UNIVERSIDAD DE LA CORUÑA

Escuela Politécnica Superior. Ferrol

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Título:

SISTEMA DE NAVEGACIÓN AUTÓNOMA PARA UN CUADRICÓPTERO

Autor:

D. ADRIÁN LÓPEZ CANOSA

Tutor:

D. FRANCISCO JAVIER BELLAS BOUZA

Directores:

D. ABRAHAM PRIETO GARCÍA

D. PEDRO TRUEBA MARTÍNEZ

Fecha:

FEBRERO 2015

Índice de contenidos

1. Introducción

El objetivo de este capítulo es aportar una visión global sobre el campo de la navegación autónoma con cuadricópteros para enmarcar adecuadamente este trabajo fin de grado. Primeramente se detalla cuál ha sido la motivación que ha llevado a emprender este trabajo, describiéndose qué es un cuadricóptero, cuáles son sus principales áreas de aplicación en la actualidad y en que consiste el problema de la navegación autónoma. A continuación, se presenta el estado del arte del campo de la navegación autónoma con cuadricópteros y donde se enmarcaría el presente trabajo dentro de las distintas líneas de investigación que hay abiertas en la actualidad.

1.1 Motivación

En los últimos tiempos los Vehículos Aéreos en Miniatura o MAV´s están teniendo un protagonismo creciente en el mundo de la tecnología. Dentro de ese amplio campo, nuestro interés se dirige hacia un tipo concreto de MAV: los cuadricópteros. El concepto de una nave voladora con cuatro rotores horizontales fue propuesto en 1922 por George de Bothezat, pero fue abandonado y sustituido por el diseño más común hoy en día, basado en dos rotores y correspondiente a lo que todos conocemos como helicóptero. La razón de que esto ocurriese es fundamentalmente que el cuadricóptero es más inestable y difícil de controlar sin un sistema avanzado de control electrónico, además de ser menos eficiente energéticamente que los helicópteros tradicionales. Sin embargo, presenta también una serie de claras ventajas con respecto al diseño imperante del helicóptero, entre las que destacan el tener un diseño mecánico mucho más simple y el hecho de tener cuatro motores, lo que nos permite reducir el tamaño de hélice para conseguir un mismo empuje.

Esta última característica también conlleva una reducción del tamaño de los propulsores, siendo así factible rodearlos con una carcasa protectora que evite impactos en vuelos en interiores, donde suele haber numerosos obstáculos. El hecho de que sean más pequeños también provoca que almacenen menos energía cinética, y por tanto, que si el rotor impacta, por ejemplo, con la mano de una persona por accidente, no le provoque daños serios.

Actualmente se les está dando a los cuadricópteros múltiples usos, tanto en el ámbito civil como en el militar. Algunas de las principales aplicaciones que ya se han comenzado a desarrollar para estos vehículos son:

- Misiones de búsqueda y rescate: el uso de una flota de cuadricópteros puede monitorizar una determinada zona y localizar el punto conflictivo de forma que se pueda enviar directamente un equipo a la zona del accidente, ahorrando

muchos recursos. Un ejemplo son los puertos de montaña, donde un esquiador puede quedar sepultado debido a una avalancha de nieve.

- Inspección de edificios tras alguna catástrofe natural: de nuevo se puede emplear un equipo de cuadricópteros para la inspección de edificios en los que hay altas posibilidades de derrumbe, por ejemplo, para buscar víctimas tras un terremoto. El uso de este tipo de vehículos tiene una clara ventaja respecto a vehículos terrestres, que tendrían grandes dificultades de circulación debido a los escombros en el suelo, o al empleo de personas, ya que pondríamos en riesgo su vida.

- Tareas de mapeado: puede utilizarse la cámara de los cuadricópteros para mapear ciertas zonas de un edificio de forma rápida y precisa, sin requerir por ejemplo de una persona que vaya tomando medidas manualmente, y obtener así mapas 3D con los que trabajar posteriormente. Esto resulta especialmente interesante en el ámbito de la Arquitectura.

- Transporte de mercancías: Amazon ha mostrado durante el año 2014 un prototipo para el transporte de productos desde sus almacenes hasta la casa del cliente (ver figura 1). Se trata simplemente de una demostración conceptual de que esto es posible, pero todavía hay mucho trabajo que realizar en ese aspecto hasta conseguir un sistema plenamente funcional y eficiente. De todos modos resulta una aplicación muy interesante debido a la rapidez con la que se pueden realizar los desplazamientos aéreos en contraste al transporte terrestre, que esta sujeto a circunstancias imprevisibles tales como los atascos.

- Otras posibles aplicaciones de uso más común pueden ser la inspección del tejado de nuestras casas, en caso de que por ejemplo haya habido una tormenta y queramos saber si hay daños en el techo; la inspección de puentes, que tienen que ser examinados anualmente y que requieren de un técnico que haga una inspección visual del estado de los pilares, lo cual para sitios de difícil acceso podría hacerse con este tipo de vehículos, o incluso en tareas de siembra, pudiendo emplearse para lanzar semillas sobre un terreno previamente preparado o bien para liberar alguna sustancia insecticida y tratar plagas de forma más sencilla.

Está claro pues que existen multitud de usos potenciales de este tipo de vehículos y por lo tanto el tema tiene un claro interés comercial de cara al futuro. Sin embargo, la mayor parte de las aplicaciones mencionadas requerirían del uso de un piloto experto que controlase el cuadricóptero, lo cual obviamente limita su campo de aplicación. Es por ello que este trabajo, basándose en la investigación que se ha venido realizando en este campo en los últimos años, trata de dar respuesta a la pregunta de cómo podemos

realizar una *navegación autónoma* con un cuadricóptero. En lo que sigue se tratará de dar una explicación clara de este concepto, ya que sobre él versa todo este trabajo.

Figura 1. *Ejemplo de un cuadricóptero empleado por Amazon para demostrar la viabilidad del transporte de mercancías empleando este tipo de vehículos [27].*

El problema de la navegación autónoma se ha estudiado profusamente en el campo de la robótica autónoma. Es por ello que debemos definir el concepto de que un robot sea completamente *autónomo*. El emplear este término implica que el robot tiene la capacidad de obtener información sobre el entorno y trabajar durante un período prolongado de tiempo sin intervención humana. Para ello resulta esencial que el robot sea capaz de obtener información del mundo real que lo rodea, por lo que es necesario disponer de *sensores*, que pueden ser de muy diferente índole: láser, ultrasonidos, cámaras de video, acelerómetros, etc. También tiene que ser capaz de interactuar con este entorno, para lo que se necesitan los *actuadores* (fundamentalmente motores) y los *efectores* (luces, brazos, ruedas, etc). Además, se necesita de un *sistema de control* (control inteligente) que es el que dota realmente de autonomía al robot. Este sistema utiliza una serie de algoritmos, métodos, etc. sobre la información del entorno proporcionada por los sensores para obtener una salida que aplicada sobre los actuadores nos permite lograr un determinado fin (por ejemplo que el cuadricóptero vuele a una determinada posición).

Con el término *navegación autónoma* se busca referirse a la forma en que un robot logra encontrar su camino dentro de un determinado entorno, es decir, el como es capaz de partir desde un determinado punto de origen a un punto de destino sin intervención externa. El concepto de navegación puede tener una connotación "global" cuando nos referimos al movimiento del robot en su totalidad de un punto a otro del espacio, o "particular" cuando nos referimos a una parte concreta del robot, por ejemplo, un brazo que queremos llevar a una determinada posición.

El problema es que típicamente el robot desconoce por completo donde se encuentra, lo que hace muy difícil el saber como llegar a su próximo destino, especialmente si ese punto no cae dentro de su rango sensorial. Para entender mejor el problema de la navegación autónoma vamos a considerar distintas situaciones que se pueden presentar cuando queremos que un determinado robot alcance un determinado objeto, por ejemplo, una pelota:

- *Supuesto 1*.

El robot tiene un mapa del mundo que le permite conocer donde se encuentra la pelota, y que además conoce donde se encuentra él mismo dentro de ese mapa. Lo que queda por hacer para alcanzar la pelota es elaborar un plan del camino a seguir entre la localización actual del robot y el objetivo y seguir dicha planificación. Esto se conoce como el *problema de planificación de ruta*.

Normalmente siempre van a existir muchos caminos posibles entre el punto de partida y el objetivo, y encontrarlos implica realizar una búsqueda en el mapa. Para hacer esta búsqueda de manera eficiente desde el punto de vista computacional se procede a convertir el mapa en un grafo, es decir, un conjunto de nodos (puntos) y líneas que los conectan. Esto se hace así porque ya existen algoritmos muy eficientes aplicados a este tipo de problemas. Por lo general se buscará el camino óptimo según algún criterio, como por ejemplo, que la distancia entre el origen y el destino sea mínima.

- *Supuesto 2*.

En este caso, el robot tiene un mapa del mundo, sabe donde se encuentra la pelota dentro de ese mapa pero desconoce donde está él mismo. Está claro que lo primero que tiene que hacer el robot es encontrarse a si mismo en el mapa. Esto se conoce como el *problema de localización*. El concepto de localización se refiere al proceso mediante el cual el robot es capaz de averiguar donde se encuentra con respecto a un determinado modelo del entorno, usando para ello todas las mediciones de los sensores disponibles. A medida que el robot se desplaza, la estimación de su posición cambia, pudiendo aparecer deriva, por lo que se trata de una tarea compleja que requiere de un cómputo activo de la posición.

Una de las formas típicas de mantenerte localizado es a través del uso de la *odometría*. Esta técnica permite al robot realizar un seguimiento de su posición de forma relativa a un punto de inicio o referencia. Para ello, usa diferentes sensores, como pueden ser la medición del número de vueltas de las ruedas del robot o como es común en los cuadricópteros, el uso de unos acelerómetros cuyas lecturas se van integrando en el tiempo. Sin embargo, hay que destacar que cuanto más lejos vaya el robot con respecto al punto de referencia, y cuantos más giros y cambios de posición haga, más impreciso se hará el proceso de odometría. Esto se debe a que cualquier tipo de medición de un

sistema físico es imprecisa y presenta ruido, por lo que cualquier pequeño error en el proceso se acumulará y crecerá con el tiempo. Es por ello clave para realizar una buena localización el uso de marcadores con los que obtener referencias que nos permitan reducir al máximo la deriva. Existen múltiples tipos de marcadores, pudiendo estos ser naturales (por ejemplo, un sistema de localización del que se hablará más adelante conocido como PTAM reconoce puntos característicos que posee el entorno de forma natural) o artificiales (cuando están puestos de forma intencionada en el entorno para ayudar al proceso de localización, como ocurre con los April Tags, sistema que también se comentará más adelante).

De modo más formal, el problema de la localización se trata como un problema de estimación de estado, que consiste en el proceso de calcular el estado de un sistema (que en el caso de la localización será por ejemplo la posición del robot) a partir de las medidas que haya disponibles (procedentes de los sensores). Las principales dificultades que surgen al abordar este problema son:

a. El proceso de estimación de estado es indirecto.

Esto quiere decir que en muchos casos no es posible medir directamente el valor del estado. Un claro ejemplo es el caso de la odometría, donde en lugar de medirse directamente la posición se hace una estimación indirecta, por ejemplo, integrando en el tiempo las lecturas de los acelerómetros.

b. Las mediciones tienen ruido.

Como ya se ha comentado, estamos tomando medidas de un de un sistema físico, con sensores reales donde los errores son inherentes al propio proceso de medición.

c. Las mediciones pueden no estar disponibles todo el tiempo.

Si lo están el proceso de estimación de la posición se llevaría a cabo en continuo (suele ser lo común en odometrías). Sin embargo lo más común es que esto no ocurra así, por lo que se suelen procesar los datos por lotes, cuando se ha conseguido recolectar la suficiente información (por ejemplo la visualización de marcadores suele ser intermitente).

Una vez resuelto este problema simplemente tenemos un problema del tipo planificación de ruta.

- *Supuesto 3.*

Ahora, el robot tiene un mapa del mundo y sabe donde se encuentra en ese mapa, pero desconoce la ubicación de la pelota en el mismo. En este caso, el robot lo que va a tener que hacer es un proceso de búsqueda hasta dar con la pelota. Además, el hecho de tener un mapa permite usarlo para planear una buena estrategia de búsqueda, una que cubra

todos los rincones del mapa y asegure que acabaremos encontrando la pelota. Esto se conoce como el *problema de cobertura*. Se suelen utilizar algoritmos heurísticos que lo que hacen es tener en cuenta criterios que pueden reducir mucho el campo de búsqueda (por ejemplo, buscar solo en los rincones de una sala porque es muy probable que el objeto se encuentre en ellos).

- *Supuesto 4*.

Se supone en esta ocasión que el robot carece de un mapa del mundo. En este caso, necesitará ir creando un mapa a medida que avanza: esto se conoce como el *problema del mapeado*. Es importante tener en cuenta que el no tener un mapa no implica necesariamente que el robot desconozca donde está: posee sensores locales que pueden decirle donde está si su localización puede ser caracterizada de forma única (sería lo equivalente en la realidad a estar al lado de algún sitio representativo, como por ejemplo, decir que estoy al lado de la Torre Eiffel). Sin embargo, lo que está claro es que sin un mapa no serás capaz de llegar nunca al museo del Louvre. En muchos casos reales sin embargo, es común que el robot desconozca totalmente donde está y tenga que buscar la pelota, por lo que tiene que resolver dos problemas: el de localización y el de cobertura.

- *Supuesto 5*.

En este último supuesto, se considera que el robot no tiene mapa del mundo y además desconoce donde está. Supongamos que para solventar estas dificultades escoge construir un mapa del mundo a medida que avanza tratando de localizarse a si mismo y buscar la pelota. Esto es lo que se conoce como *problema de localización y mapeado simultáneo (SLAM – Simultaneous Localization and Mapping)* y es posiblemente el caso más complejo, ya que es, como reza la expresión popular, "la pescadilla que se muerde la cola": para crear un mapa, necesitas saber donde estás, pero para saber donde estás necesitas tener un mapa. El SLAM realiza ambas cosas al mismo tiempo. La principal dificultad que hay que sortear cuando se trabaja en SLAM es la ambigüedad que aparece cuando tenemos múltiples lugares que son prácticamente idénticos. Esto se conoce con el nombre de *problema de asociación de datos*. Puede entenderse fácilmente si uno se imagina estar en una gran ciudad y trata de orientarse fijándose exclusivamente en calles donde hay cafeterías y edificios. Es muy probable que nos perdamos porque nos costará mucho diferenciar una calle de otra. Con esto se pretende dar a entender que no es un problema trivial y presenta serias complejidades a las que los investigadores en el campo de la robótica han tratado de dar respuesta en forma de diferentes algoritmos.

Como se puede suponer, los distintos aspectos que componen el problema de la navegación autónoma han sido profusamente estudiados en el campo de la robótica en los últimos años, dando lugar a numerosos artículos de investigación y libros donde se

explican los algoritmos desarrollados en este campo. Sin embargo, se trata de uno de los problemas todavía abiertos en robótica, y la dificultad que entraña hace que cada año se publiquen artículos que van engrosando el cuerpo de trabajo sobre la materia.

1.2 Trabajos previos

Como ya se ha manifestado, este trabajo trata de dar respuesta al problema que plantea la navegación autónoma de un tipo concreto de robot: el cuadricóptero. La investigación dentro de este campo puede dividirse en diferentes áreas: por un lado tenemos los trabajos cuyo principal objetivo es desarrollar un sistema preciso y rápido de control del cuadricóptero utilizando un sistema externo de captura de movimiento. Este enfoque permite realizar maniobras tremendamente complejas ya que se puede estimar con precisión milimétrica la posición del cuadricóptero en el espacio, pero presenta varios inconvenientes, como que limita la navegación al espacio de laboratorio que cubren los sensores de captura o el elevado precio de los mismos (del orden de miles de euros). Algunas investigaciones que podemos destacar en este ámbito han estudiado el uso de cuadricópteros en la realización de maniobras agresivas en espacios reducidos [1], en tareas de construcción colaborativa [2], en maniobras consistentes en el lanzamiento y recogida de pelotas empleando una red [3], o en la coordinación de grandes flotas de estos robots [4].

Por otro lado, están las líneas investigación que buscan desarrollar una alternativa al caro y limitado sistema de sensorización externa, que es donde se enmarca el presente trabajo. Estas investigaciones han tratado de resolver el problema que supone la navegación en interiores y en entornos desconocidos, donde no es posible usar el GPS, un sistema que simplifica mucho la navegación en exteriores. Por el contrario, en interiores solo vamos a disponer de los sensores de abordo del cuadricóptero, que no proporcionan referencias absolutas como el GPS. Se han realizado múltiples trabajos tratando de resolver el problema: usando sensores láser [5], usando cámaras RGB-D del tipo Kinect [6] o usando cámaras estereoscópicas [7]. El principal problema de los citados enfoques es que, aunque permiten realizar una estimación precisa y con una escala absoluta del entorno, tienen un alto precio y son sensores muy pesados y difíciles de incorporar al cuadricóptero, además de consumir mucha energía, lo que limita mucho la duración de la batería.

Es por ello que los sensores más utilizados para estimar la posición son la unidad inercial de medida (IMU – Inertial Measurement Unit) y la cámara monocular. Con estos sensores, la navegación se basa en realizar una odometría con las lecturas de los acelerómetros de la IMU integradas a lo largo del tiempo y las estimaciones de la velocidad resultantes de la aplicación de algoritmos de flujo óptico a las imágenes de la cámara como proponen trabajos tales como [8]. Sin embargo estos sistemas funcionan bajo ciertas suposiciones como suelos planos, horizontales, bien texturizados, etc, que

no siempre se cumplen, y lo más importante, que están sujetos a una importante deriva con el tiempo, por lo que no son adecuados para una navegación autónoma de larga duración. Se necesita pues integrar alguna solución que nos permita eliminar esta deriva. Algunas de las soluciones planteadas han sido hacer que el cuadricóptero aprenda un mapa offline de un vuelo anterior realizado manualmente, de modo que pueda repetir posteriormente la misma trayectoria sin ningún tipo de desviación [9] el uso de marcadores artificiales [10] o mediante diversos planteamientos del tipo SLAM monocular [11][12][13].

El presente trabajo fin de grado se centra en resolver el problema de la navegación autónoma en cuadricópteros para entornos de interior con un condicionante a mayores: la utilización de un modelo comercial en su configuración estándar. Así, la plataforma empleada será un cuadricóptero comercializado por la empresa Parrot desde 2010 a un precio de unos 300 euros como juguete de alta tecnología y conocido con el nombre de AR Drone. Su interés para investigaciones en el ámbito de la robótica se ha discutido en algunas publicaciones tales como [14][15]. La realización de una navegación autónoma con un hardware de bajo precio hace que la tarea sea más complicada, debido a la imprecisión de los actuadores, elevado ruido en las mediciones de los sensores, importantes retrasos en el intercambio de información entre sistema central (ordenador personal) y el cuadricóptero y a los limitados recursos de computación de abordo. En contrapartida, los desarrollos aquí realizados con una herramienta tan popular serán mucho más difundibles y útiles para la comunidad técnica.

Uno de los únicos trabajos que ha realizado una investigación completa sobre el SLAM en un cuadricóptero AR Drone es [16]. Esta investigación se ha tomado como punto de partida para este trabajo, ya que está disponible de forma íntegra en forma de software libre para la plataforma ROS (Robot Operating System), lo que facilita su uso para posteriores proyectos de investigación como el nuestro, que trata de adaptar el sistema a unas necesidades específicas, resultando sencillo el realizar aportes y modificaciones sobre el software de partida. Como se detallará más adelante, este sistema presenta una serie de deficiencias a la hora de navegar en base a marcadores naturales. En este sentido, una de las contribuciones más novedosas que se han realizado en el presente trabajo fin de grado es el añadir un sistema de marcadores artificiales conocidos como April Tags [17], que presentan un gran potencial en tareas que requieren una gran precisión en la localización (proporcionan de forma precisa la posición del April Tag con respecto a la cámara) y que a día de hoy solo se les ha dado uso en la calibración de cámaras [18], pero no como herramientas para la navegación autónoma.

2. Objetivos

El objetivo fundamental de este proyecto es *lograr que el cuadricóptero comercial AR Drone navegue de forma autónoma, es decir, que empleando la información procedente de sus sensores de abordo sea capaz de localizarse a sí mismo en el espacio y de realizar unas maniobras preestablecidas sin tener que intervenir en absoluto*.

Para ello se ha partido del estudio de un sistema propuesto por el Grupo de Visión Artificial de la Universidad Técnica de Múnich (Technical University of Munich o también conocida por sus siglas TUM) especialmente desarrollado para el AR Drone y distribuido libremente bajo licencia GNU para su uso en investigación. Dicho sistema proporciona una herramienta básica con la que estimar la posición del dron a partir de la información enviada por la cámara (empleando unos algoritmos de monocular SLAM conocidos como PTAM) y los sensores onboard (IMU, altímetro, etc) del mismo mediante un filtro de Kalman. Además, incorpora también un controlador PID con el que enviar órdenes al cuadricóptero para que se posicione en un determinado punto en base a la estimación de estado proporcionada por el filtro. Este sistema pone a nuestra disposición unas herramientas básicas con las que navegar, permitiéndonos una localización relativamente precisa del cuadricóptero y el poder volar patrones sencillos, como por ejemplo un cuadrado de un metro de lado. Este TFG se han centrado en generalizar esta navegación, aumentando la precisión de la localización, especialmente en la estimación de la orientación del cuadricóptero, y permitiendo con ello la realización de maniobras más complejas. Para ello se ha utilizado un sistema de marcadores artificiales conocido como April Tags Fiducial System. Los subobjetivos fundamentales a alcanzar en este proyecto son, pues, los siguientes:

- *Estudiar en profundidad el paquete de software publicado por la Universidad Técnica de Múnich con el objetivo de poder realizar una navegación básica con el AR Drone*.

- *Implementar los algoritmos necesarios para optimizar el procesado de los datos procedentes de los sensores onboard de AR Drone*.

- *Desarrollo de un sistema de localización para el AR Drone en base al sistema de marcadores artificiales invariantes conocido como April Tags Fiducial System*.

- *Proporcionar a los investigadores del GII una librería de control para el AR Drone que utilice el April Tags Fiducial System y sea fácilmente integrable con otras librerías existentes (ROS)*.

3. Fundamentos teóricos

El objetivo de este capítulo es presentar formalmente los distintos elementos de los que se compone el sistema que se ha diseñado en este trabajo fin de grado. En primer lugar, se aportará una visión general del mismo en el apartado 3.1, presentando los distintos elementos que lo componen. Posteriormente, se describirán en mayor detalle cada uno de los bloques básicos sobre los que se fundamentan las aportaciones de este trabajo, a saber, el sistema operativo para robots ROS en el apartado 3.2, el cuadricóptero comercial AR Drone en el apartado 3.3, el paquete de software TUM AR Drone en el apartado 3.4, y el sistema de marcadores artificiales April Tags Fiducial System en el apartado 3.5.

3.1 Visión general del sistema

En este apartado se presentará de forma general el sistema de navegación diseñado en este trabajo fin de grado. Para ello, se explicará detalladamente el esquema que se ha representado en la figura 2, en el cuál se ilustra la totalidad del mismo.

En primer lugar se debe hablar del AR Drone, cuadricóptero con el que se pretende llevar a cabo una navegación autónoma. El AR Drone cuenta con una serie de sensores dentro de los que se hace una distinción entre dos unidades: los sensores onboard y la cámara frontal. La primera de ellas está compuesta a su vez por una IMU (Unidad Inercial de Medida), que cuenta con un acelerómetro y un giroscopio de tres ejes; un altímetro, un barómetro y una cámara inferior, denominada así porque está situada en la parte de abajo del cuadricóptero apuntando al suelo. Esta cámara obtiene un flujo de imagen que es procesado por el propio firmware del AR Drone, mediante la aplicación de algoritmos de flujo óptico, para obtener una estimación de las velocidades lineales. La cámara frontal se ha separado del resto de sensores debido a que se busca hacer hincapié en que a los frames de imagen obtenidos con esta cámara, se le aplica un procesado posterior en el ordenador totalmente diferente del que realiza el firmware para obtener las velocidades lineales con los frames procedentes de la cámara inferior.

Existen pues dos conexiones que salen del bloque de sensores: por un lado el envío de información sobre el estado del dron en el entorno (velocidad, aceleración, cambio de orientación con el tiempo, altura con respecto al suelo, etc), y por otro lado el envío de los frames de imagen captados por la cámara. Todos estos datos son gestionados por el firmware del cuadricóptero con el fin de hacer posible el vuelo cuando se emplea la aplicación oficial de la empresa desarrolladora.

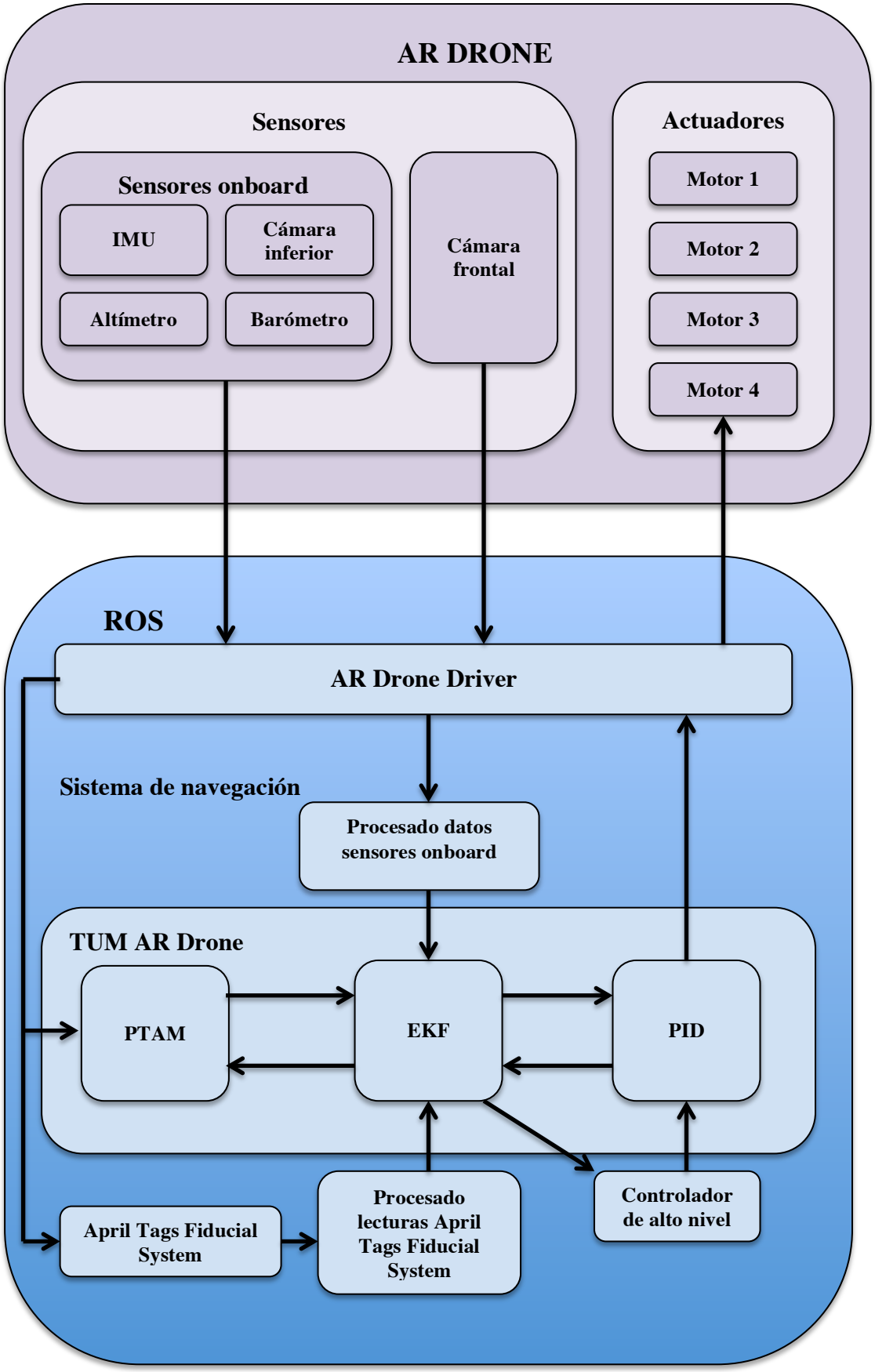


Figura 2. *Esquema general de funcionamiento del sistema de navegación autónoma.*

No obstante, desde el punto de vista de este trabajo fin de grado, interesa tener acceso a estos datos y poder enviarle órdenes al cuadricóptero sin tener que depender de la aplicación oficial, que nos limita mucho las opciones de control del dron. Esto es posible gracias a que el AR Drone crea una wireless LAN a través de la que envía los datos recogidos por los sensores, por lo que si nos conectamos con nuestro ordenador personal al cuadricóptero, deberíamos tener acceso a los mismos. Sin embargo, la información enviada por el AR Drone tiene que ser traducida a un formato que podamos utilizar, y es en este punto donde entra en escena un sistema operativo para robots conocido como ROS.

Una serie de desarrolladores han implementado en este sistema un driver que permite traducir la información enviada por el dron al formato que maneja este sistema operativo y viceversa. Esta capa es la que hace posible trabajar fácilmente con la información procedente de los sensores del cuadricóptero, y al mismo tiempo, traducir a señales que el dron pueda entender y ejecutar los comandos de velocidad calculados por el sistema de navegación (unidades situadas por debajo de la capa del driver en el esquema de la figura 2). El principal elemento del mismo se conoce como TUM AR Drone. Este bloque es otro programa desarrollado en ROS cuyo componente fundamental es un filtro de Kalman extendido o EKF, que calcula una estimación del vector de estado del dron (posición, orientación y velocidad en un determinado instante) a partir de los datos procedentes de los sensores onboard del AR Drone tras pasar un procesado previo, y de una estimación de la posición del dron obtenida mediante la aplicación de un software de procesado de imágenes conocido como PTAM. El TUM AR Drone también cuenta con un controlador PID, que calcula los comandos de velocidad que hay que enviar al cuadricóptero para que alcance una determinada posición definida por un controlador de alto nivel. Este controlador permite que el usuario establezca una determinada ruta a seguir por el AR Drone en función de un determinado criterio (por ejemplo que el ángulo con el que el cuadricóptero vea unos determinados marcadores artificiales sea siempre lo menor posible). Para ello recibe una estimación de la posición del dron por parte del EKF, calcula los comandos de alto nivel necesarios para que el dron siga esa ruta y los envía al PID. Tanto los comandos de alto como de bajo nivel están en formato ROS, y por ello es necesario que los de bajo nivel se envíen al driver y que este los transforme al formato que maneja el AR Drone, antes de enviarlos de vuelta para que sean ejecutados por los actuadores.

Sin embargo, hay que destacar que el software TUM AR Drone limita mucho la navegación debido a que, por ejemplo, no es capaz de estimar correctamente giros del dron. Es por ello que el presente trabajo fin de grado se ha centrado en incorporar un elemento nuevo conocido como April Tags Fiducial System. Se trata de unas librerías de procesado de imágenes que se han tenido que adaptar a ROS y que calculan a partir de cada frame de imagen procedente de la cámara frontal a través del driver en formato ROS, la posición de una serie de marcadores artificiales impresos a los que llamamos

tags con respecto a un sistema de coordenadas centrado en la cámara. Las lecturas procedentes del April Tags Fiducial System se envían a un programa en el que se realizan un procesado mediante el cual es posible conocer la posición y orientación del dron, es decir, que este sistema puede considerarse como otro "sensor" del que obtenemos observaciones que añadir en el filtro de Kalman.

3.2 ROS

ROS son las siglas para Robot Operating System, y se trata de un sistema meta-operativo de código abierto que provee de los servicios básicos que se esperan de un sistema operativo destinado a trabajar con robots, incluyendo características tales como abstracción de hardware, control de dispositivos a bajo nivel, funcionalidades de uso común implementadas de serie, canales para intercambio de mensajes entre distintos procesos y gestión de paquetes entre otros. También proporciona las herramientas y librerías para obtener, compilar, escribir y ejecutar el código que implementemos.

Existen en el mercado otras plataformas similares a ROS, como pueden ser Player, Orocos, CARMEN, Orca o Microsoft Robotics Studio, y la pregunta que posiblemente se haga el lector es cuál es el motivo por el que se ha escogido ROS de entre todas ellas. Una de las características clave de ROS es que al tratarse de software libre su uso es gratuito y como usuarios tenemos total libertad para modificar las cosas que deseemos a nivel de software. Sin embargo, la mayoría de los otros programas que se han citado también se enmarcan dentro de esta categoría, por lo que la pregunta no está completamente resuelta. Los puntos fuertes de ROS se encuentran en:

- **Computación distribuida:** la mayoría de los robots actuales requieren de software que ejecutan diferentes procesos en diferentes ordenadores. También es una práctica común dividir el software del robot en pequeñas partes independientes que cooperan para lograr el objetivo propuesto. Se entiende pues que es necesario un canal de comunicación entre los múltiples procesos que están teniendo lugar en diferentes o incluso un mismo ordenador. ROS nos proporciona herramientas que hacen muy sencillo implementar este tipo de comunicación en nuestros programas.

- **Reutilización de software:** el vertiginoso progreso que ha experimentado el campo de la robótica en los últimos años ha dado como resultado un buen número de algoritmos para resolver algunas de las tareas típicas como la navegación, planificación de tareas, creación de mapas, etc. Esta claro que las existencia de estos algoritmos solo es útil si hay algún modo de aplicarlos a nuevos contextos sin necesidad de reimplementarlos de manera particular para cada caso. ROS proporciona funcionalidades que nos permiten evitar este tipo de problemas:

1. Incorpora paquetes donde ya vienen implementadas versiones estables y debugeadas de muchos algoritmos importantes en robótica.
2. No impone restricciones al código desarrollado en su utilización en otros frameworks tales como OpenRAVE, Orocos o Player.
3. Proporciona librerías "agnósticas" con limpias interfaces funcionales.
4. Independencia del lenguaje de programación utilizado: ROS está actualmente implementado en Python, C++ y Lisp, existiendo librerías experimentales en Java y Lua.

- **Facilidad de la realización de pruebas:** una de las razones por las que el desarrollo de software para robots es a menudo más difícil que otros tipos de desarrollos es que la realización de pruebas consume mucho tiempo y es frecuente que aparezcan errores, por no hablar de que en muchos casos no se dispone de un robot físico en el que realizar las pruebas. ROS nos proporciona diferentes alternativas de abordar el problema:

  1. ROS separa los sistemas de bajo nivel encargados del control del hardware de los de alto nivel, enfocado más bien al procesado y a la toma de decisiones. Gracias a esta separación, podemos reemplazar estos programas de bajo nivel y el software correspondiente por un simulador, con lo que nos podemos centrar en probar la parte de alto nivel del sistema.
  2. ROS también nos proporciona una forma sencilla de grabar y volver a reproducir datos de los sensores u otro tipo de mensajes. Esto nos permite aprovechar más eficientemente el tiempo invertido en la realización de pruebas que si trabajásemos directamente tomando medidas del robot físico. Además, al quedar grabados podemos reproducirlos las veces que queramos, y ver como quedan afectados al cambiar algún parámetro del programa.

- **Número de usuarios**: posiblemente el principal punto fuerte de ROS sea la gran cantidad de personas del ámbito de la robótica que lo utilizan, lo que implica que tiene una amplia "masa crítica" que se encarga de actualizar constantemente los repositorios, poner a disposición de cualquier persona nuevos algoritmos, funcionalidades,etc. fruto de algún proceso de investigación, corregir cualquier bug o malfuncionalidad que pudiese surgir en la plataforma, actualización constante de la misma para que sea posible trabajar con el hardware más puntero, etc.

Actualmente ROS solo funciona en plataformas basadas en UNIX. El software de ROS se utiliza fundamentalmente en Ubuntu, aunque los diversos usuarios también han estado realizando contribuciones en otras plataformas Linux como Fedora o Gentoo, así

como en Mac OS. También existe la posibilidad de crear un puerto para trabajar en Microsoft Windows, aunque no ha sido muy explorada.

Es importante dejar claro que ROS no es ni un lenguaje de programación, ni solamente una librería (esta es solo una de sus múltiples funcionalidades, incorporando también un servidor central, herramientas a través de línea de comandos, herramientas gráficas, sistema de compilación, etc.), ni tampoco es un entorno integrado de desarrollo (IDE). Para mayor información sobre los detalles técnicos de ROS consultar el anexo I.

3.3 AR Drone

El cuadricóptero escogido para la realización de este trabajo fin de grado es el AR Drone (ver figura 3), desarrollado por la empresa francesa Parrot, cuya primera versión fue presentada en el año 2010 como un diseño de alta tecnología para juegos de realidad aumentada. La versión comercial está pensada para ser controlada por un piloto humano empleando un smartphone, una tablet o un PC.

Figura 3. *Fotografía del Parrot AR Drone [31].*

Sin embargo, poco después de su lanzamiento comenzó a atraer la atención de universidades y centros de investigación de todo el mundo, y hoy en día es el soporte de múltiples proyectos en los campos de inteligencia artificial, robótica o visión por ordenador en múltiples universidades del mundo tales como el MIT (Massachusetts Institute of Technology) o la ETH (Eidgenössische Technische Hochschule Zürich). Los principales motivos de su popularidad en usos de investigación son su bajo precio (de unos 300 euros la unidad), el acceso sencillo a los datos de los sensores, y la robustez y facilidad de vuelo que posee.

3.3.1 Descripción de los fundamentos mecánicos de su movimiento

El cuadricóptero es un vehículo aéreo que posee cuatro rotores que permiten realizar diversas maniobras con el mismo: despegar, avanzar hacia adelante, hacia los lados, etc. Los cuatro rotores al girar generan un empuje que permite que el cuadricóptero se

acelere y se desplace. Una consideración muy importante es que hay dos rotores que giran en sentido horario (el 2 y el 3) y otros dos que giran en sentido antihorario (rotores 1 y 4), lo que permite que los pares que generan se anulen, de modo que si tenemos los cuatro motores girando a la misma velocidad, el cuadricóptero se mantendrá en una posición determinada en el aire sin moverse: el empuje generado por los rotores compensa la fuerza gravitatoria y al girar todos a la misma velocidad, el par total es cero, y por tanto, no hay giro. Las distintas maniobras básicas que puede llevar a cabo el dron (obviamente también pueden ejecutarse distintas combinaciones de estas maniobras básicas) se comentan a continuación:

- Desplazamiento en vertical ascendente (caso 2 en la figura 4) o descendente (caso 3 en la figura 4): se consigue incrementando o disminuyendo la velocidad de los cuatro rotores en la misma cantidad (se incrementa o disminuye la fuerza de empuje (*thrust* en inglés)). Es importante conocer que si se desea mantener la posición vertical, la velocidad de giro tiene que ser tal que el empuje compense la fuerza gravitatoria (caso 1 en la figura 4).
- La rotación con respecto al eje z (es decir rotación en el yaw) se consigue aumentando la velocidad de los motores 1 y 4 y disminuyendo la velocidad de los motores 2 y 3 si queremos crear un par en sentido horario (caso 5 en figura 4) o a la inversa si queremos un par en antihorario (caso 4 en la figura 4).
- El movimiento horizontal puede conseguirse incrementando la velocidad de un motor mientras se disminuye la velocidad del opuesto, dando lugar a un cambio en el roll o el pitch, y consecuentemente, dando lugar a que la fuerza de empuje no esté alineada con el eje z y se genere, por tanto, una aceleración en el plano XY. El caso 6 de la figura 4 se correspondería con un desplazamiento hacia adelante, mientras que el caso 7 representaría un desplazamiento hacia atrás. El caso 8 se relaciona con un desplazamiento hacia la izquierda y el caso 9 con un desplazamiento hacia la derecha.

3.3.2 Descripción de características técnicas

El modelo concreto que se ha utilizado es el AR Drone 2.0 que tiene unas dimensiones de 51,7 cm x 51,5 cm con la carcasa de interiores y unos 45,1 cm x 45,1 cm con la carcasa de exteriores. Tiene cuatro rotores de 20 cm de diámetro, unidos a un robusto esqueleto de fibra de carbono en forma de cruz que da estabilidad al conjunto. La carcasa de interiores (ver figura 5.a) es de poliestireno y está pensada para proteger al dron de choques contra diversos tipos de obstáculos (muebles, puertas, muros, etc.), que podrían causar serios daños a los rotores.

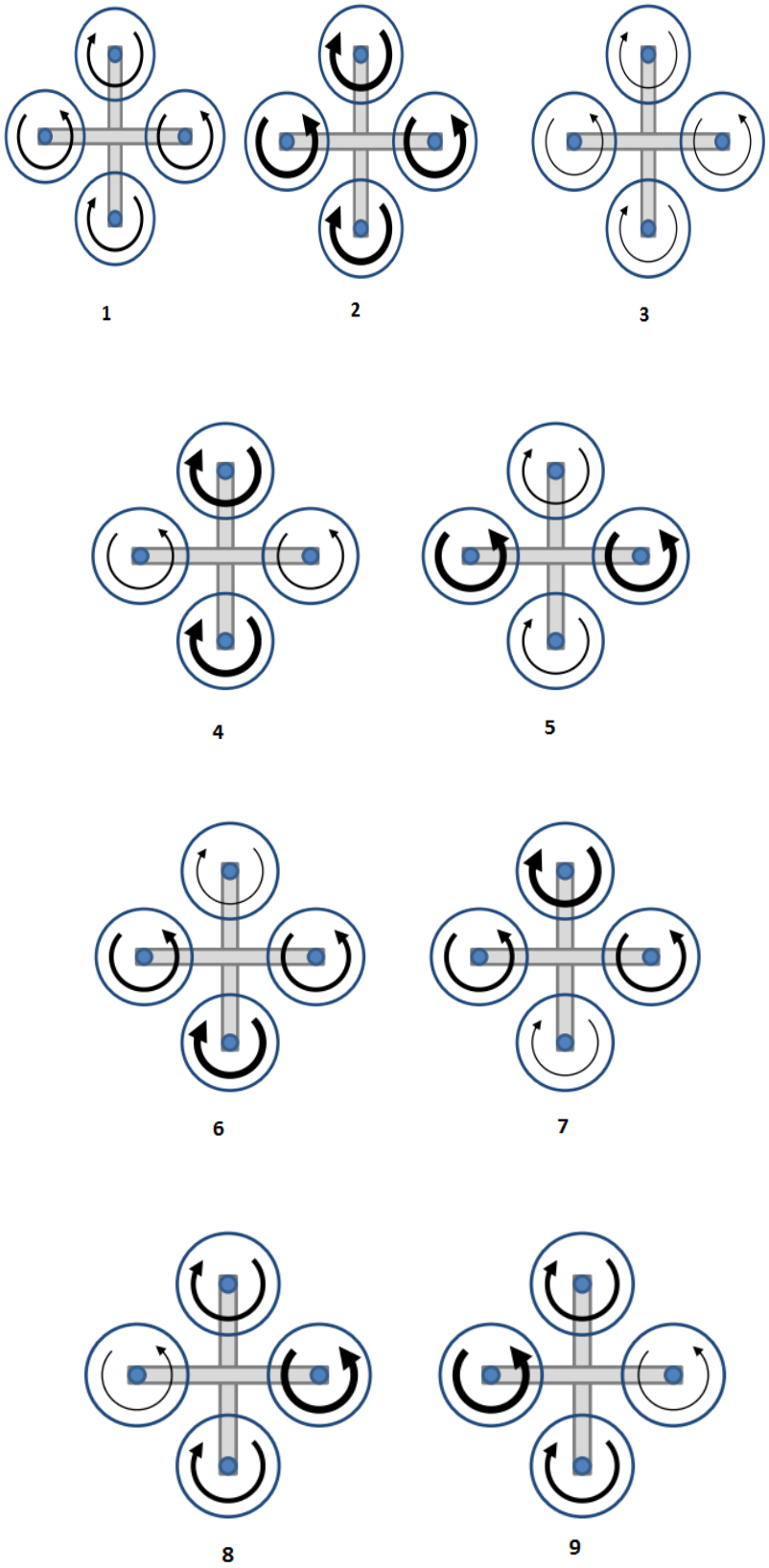


Figura 4. *Representación esquemática de la configuración de los motores para las distintas maniobras de vuelo [29].*

La carcasa de exteriores (ver figura 5.b) carece de la estructura de protección de los rotores ya que se necesita una mejor maniobrabilidad y alcanzar mayores velocidades, y es menos probable que se produzcan colisiones peligrosas. El AR Drone 2.0 pesa 380 g con la carcasa de exteriores y 420 g con la de interiores.

(a) *(b)*

***Figura 5**. Fotografía que muestra el aspecto del AR Drone 2.0 con la carcasa de interiores (a) y con la de exteriores (b) [29].*

El dron también está equipado con dos cámaras: una orientada hacia adelante, siendo esta una cámara de alta definición, con una resolución de 1280x720 pixeles (aunque para nosotros realmente se reduzca a 640 x 360 pixeles debido a los problemas que surgen con el firmware en la decodificación de imagen) que funciona a 30 fps y con una lente gran angular (92º diagonal) y otra hacia abajo, siendo esta una cámara QVGA vertical (320 x 240 pixeles) a 60 fps con un campo de visión de 64º. El software de abordo usa esta cámara para estimar la velocidad horizontal del dron. Es importante apuntar que solo uno de los dos streams de video puede ser mostrado en el ordenador en un mismo instante.

El AR Drone también cuenta con un altímetro por ultrasonidos, un medidor de presión de ±10 Pa de precisión, y una IMU (Inertial Measurement Unit o Unidad de Medición Inercial) que consta de un acelerómetro de tres ejes (mide las aceleraciones en x, y, z) de ±50 mg de precisión, un giroscopio de tres ejes (mide las variaciones en los ángulos del roll, el pitch y el yaw con respecto al tiempo) con una precisión de 2000 º/s y un magnetómetro de tres ejes con una precisión de 6 grados. El controlador de abordo está compuesto por un procesador ARM Cortex A8 de 1 GHz y 32 bits, una RAM DDR2 de 1 Gbit y 200 MHz que funciona bajo la distribución Linux 2.6.32. Tiene también un puerto USB 2.0 y se controla mediante una red LAN inalámbrica.

3.4 TUM AR Drone

El estudio en materia de navegación autónoma de cuadricópteros escogido como punto de partida para este trabajo fin de grado fue propuesto por el Grupo de Visión Artificial de la Universidad Técnica de Múnich (Technical University of Munich o también conocida por sus siglas TUM), que lanzó en 2011 bajo licencia GNU un paquete de software para su uso en investigación [16]. A continuación se hará una descripción de su funcionamiento, para lo que se echará mano de nuevo del esquema representado en la figura 2.

El TUM AR Drone está compuesto por tres elementos: el fundamental es el filtro de Kalman Extendido o EKF, algoritmo diseñado para tratar con sistemas no lineales que calcula a partir de una serie de medidas sujetas a ruido que le llegan de diversas fuentes una estimación más precisa del estado del dron (esto es, su posición, orientación y velocidad en un determinado instante)**.** Una de estas fuentes de información es el PTAM, elemento que analiza los frames de video proporcionados por la cámara frontal del AR Drone mediante algoritmos de visión por ordenador, calculándose de forma simultánea un mapa del entorno 3D (mediante la detección de marcadores o puntos significativos en la imagen) y un seguimiento de la posición de la cámara. Esto quiere decir que para cada frame de video que entra en el PTAM vamos a tener una estimación de la posición de la cámara (y por tanto del AR Drone) como salida. Es importante mencionar que la comunicación es bidireccional, ya que el filtro envía a su vez una estimación del estado del dron que el PTAM usa para calcular la escala del mapa y para inicializar el proceso de seguimiento de la cámara en caso de que este se pierda.

El EKF recibe además información sobre la posición, orientación, velocidad del dron a través de los datos enviados por los sensores onboard convertidos a formato ROS mediante el driver y sometidos a un procesado que abarca la el filtrado de los distintos datos para eliminar ruido, su calibración y su sincronización (ya que cada los sensores envían información con distintas frecuencias) antes de ser introducidos en el filtro.

El TUM AR Drone también cuenta con un controlador PID que recibe estimaciones del estado del dron en cada instante de tiempo por parte del EKF a partir de las cuales calcula los comandos de control de bajo nivel que hay que enviarle al mismo para que alcance una determinada posición objetivo, la cual viene determinada por un controlador de alto nivel, cuyo funcionamiento ya se ha comentado en el apartado 3.1. Es importante comentar que el PID envía también los comandos calculados al EKF, suponiendo otra entrada de información en base a la que calcular el estado del dron.

Si se desean conocer en mayor profundidad los detalles técnicos y funcionamiento del TUM AR Drone consultar el anexo II.

3.5 April Tags Fiducial System

Los April Tags son un sistema de marcadores artificiales (ver figura 6) desarrollados en la Universidad de Michigan útiles para una gran variedad de tareas, que van desde aplicaciones de realidad aumentada, robótica y calibración de cámaras. Estos marcadores pueden obtenerse fácilmente a través de una impresora corriente. Las librerías de detección (implementadas en C y Java), puestas a disposición pública, ya que se distribuyen en calidad de software libre a través de Internet, nos permiten obtener la posición 3D, la orientación y el identificador del marcador, todo ello de forma relativa a la cámara.

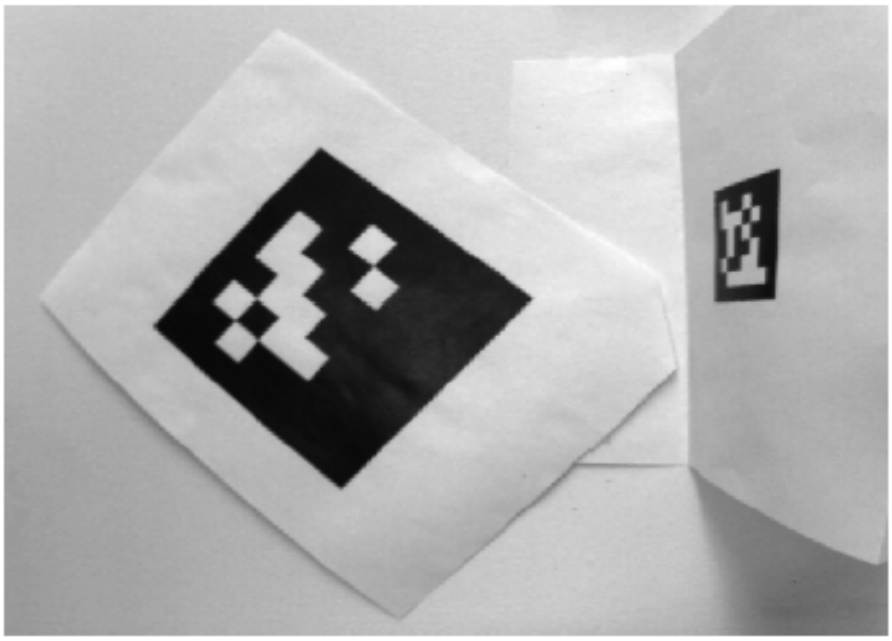


***Figura 6**. Fotografía del marcador artificial conocido como April Tag [17].*

En el ámbito de la visión por ordenador, los sistemas basados en marcadores naturales (como el PTAM), suelen ser los más interesantes ya que se basan en la capacidad del robot en detectar características de un entorno no manipulado. Sin embargo, actualmente estos sistemas tienen una serie de requerimientos para funcionar de forma adecuada que limitan mucho su aplicación en campos como el de la navegación autónoma. Es por ello que en estos ámbitos donde las tareas percepción del robot no son lo esencial (sino más bien la variedad de maniobras que se pueden llevar a cabo con precisión) que los marcadores artificiales ganan interés, por el hecho de estar especialmente diseñados para ayudar al proceso de detección y consecuentemente facilitar el proceso de localización.

Estos marcadores son parecidos a otros códigos de barras bidimensionales como pueden ser los QR, pero sin embargo presentan importantes diferencias en sus objetivos y aplicaciones. En los códigos QR, se necesita de una persona que alinee la cámara de forma correcta con el marcador, siendo este fotografiado a una resolución elevada,

obteniéndose cientos de bytes que nos pueden dar información sobre, por ejemplo, una dirección web. Sin embargo, los April Tags nos proporcionan una menor carga de información (unos 12 bits), pero está diseñado para ser detectado y localizado automáticamente, incluso cuando tenemos una baja resolución, está iluminado de forma desigual, girado de manera extraña, etc. Para ayudar a su detección a largas distancias, los April Tags están compuestos de menos celdas de datos: mientras que un marcador QR tiene unos 268 píxeles, los April Tags constan de unos 49 a 100 píxeles. Además, a diferencia otros sistemas de tipo código de barras en 2D, en los que la posición del marcador en la imagen es irrelevante, en este sistema se proporciona una posición y orientación relativa entre la cámara y el marcador. Los April Tags también son capaces de detectar múltiples marcadores en una sola imagen.

Algunas de las aplicaciones de estos marcadores son el mejorar la interacción entre humano y robot, permitiendo a la persona mandarle mensajes al robot como que le siga o que espere, mostrando un marcador determinado para cada orden. También se usan mucho en el ámbito de la investigación de percepción artificial en el benchmarking y evaluación de sistemas robóticos basados en sistemas tipo marcadores naturales como el ya presentado SLAM (Simultaneous Localization and Mapping) ya que nos permite obtener las trayectorias reales del robot en bucles de control cerrados y comparar la predicción de estos sistemas con una referencia real.

En lo que respecta a otros sistemas semejantes al April Tags podemos destacar el ARToolkit, que están entre los primeros sistemas de marcadores artificiales empleados en tareas de seguimiento de robots. Los marcadores tenían forma cuadrada y contenían la información en el interior, estando delimitada la figura por un borde negro. Una de sus principales características era que la información no estaba codificada en binario, sino que usaba símbolos como el carácter latino “A”. A la hora de decodificar el marcador, la información del mismo (sampleada a alta resolución) se correlacionaba con una base de datos de marcadores conocidos, siendo escogido el que más se parezca y enviándose la información al usuario.

Una de las mayores desventajas de este enfoque es el coste computacional asociado a la decodificación del marcador, ya que cada modelo del mismo requería por separado un lento proceso de operación. Otra desventaja es la de la dificultad de generar modelos que sean aproximadamente ortogonal el uno al otro. El esquema de detección usado por el ARToolkit se basa en una simple binarización de la imagen de entrada a partir de un determinado umbral de detección especificado por el usuario. Este esquema es muy rápido, pero no robusto a cambios de iluminación ni a que se tapen lo más mínimo los bordes del marcador.

Otro sistema, el ARTag, mejora la detección y los sistemas de codificación. El mecanismo de detección se basa en un gradiente de imagen, haciéndolo robusto a los

cambios de iluminación. Los detalles del algoritmo de detección de este sistema no son de acceso público, sin embargo, se sabe que este mecanismo es capaz de detectar marcadores cuyos bordes están parcialmente tapados. Este sistema también incluye el primer sistema de codificación basado en la corrección del error hacia adelante, lo que permite que los marcadores sean más fáciles de generar, de correlacionar y de establecer relaciones de ortogonalidad entre los marcadores.

También existen otro tipo de marcadores artificiales además de los tipo código de barras bidimensional, como pueden ser cámaras que detectan luces LED. Sin embargo, los marcadores bidimensionales presentan una clara ventaja frente a los anteriores, y es que son mucho más baratos al poder imprimirse fácilmente en un impresora corriente, y permiten realizar una localización en seis grados de libertad de la posición de la cámara sin necesidad de emplear múltiples LED.

Centrándonos ya en el funcionamiento de los April Tags, el sistema está compuesto por dos componentes fundamentales: el detector del marcador y el sistema de codificación. En primer lugar se va a describir el detector, cuyo trabajo es fundamentalmente el estimar la posible posición del marcador en la imagen. De forma muy simplificada, se podría decir que el detector trata de localizar cuatro regiones (“quads”) que tienen el lado interior más oscuro que el exterior. De hecho, los propios marcadores tienen bordes blancos y negros con el fin de facilitar este proceso.

Si se desean conocer en mayor profundidad los detalles técnicos y funcionamiento del April Tags Fiducial System consultar el anexo III.

4. Desarrollo del sistema de localización y navegación

El objetivo de este capítulo es describir las tareas que se han desarrollado en este trabajo fin de grado para conseguir que el cuadricóptero comercial AR Drone navegue de forma autónoma, es decir, que empleando la información procedente de sus sensores de abordo sea capaz de localizarse a sí mismo en el espacio y consecuentemente de realizar unas maniobras preestablecidas sin tener que intervenir en absoluto.

Para lograrlo, se decidió tomar como punto de partida un sistema de navegación para el cuadricóptero comercial AR Drone ya existente, conocido con el nombre de TUM AR Drone y que ya se presentó anteriormente en el apartado 3.4. Dicho software se sometió a numerosas pruebas y se llegó a la conclusión de que el PTAM elemento clave para la localización del AR Drone en el citado sistema de navegación, limita la navegación de forma considerable debido a varios motivos:

- *Los métodos de monocular SLAM son incapaces de distinguir la rotación pura de la traslación*. Esto se debe a que la posición de la cámara se estima en base a la triangularización por geometría epipolar de los puntos característicos pertenecientes a, al menos, dos imágenes diferentes. Cuando, por ejemplo, mantenemos el dron en las mismas coordenadas del espacio pero hacemos un giro en el eje z de 45º, los puntos característicos de la imagen antes del giro y después del mismo van a verse en distintas posiciones, por lo que los algoritmos de triangularización establecerán que el dron se ha desplazado, cuando no es así. La única forma de girar sin que falle la estimación es realizar el giro acompañado de una traslación suficiente, pero en muchas ocasiones nos interesará girar exclusivamente, lo que supone una limitación seria en la navegación.

- *La estimación de la posición en los métodos de monocular SLAM se da en función de la escala*, y esta no puede ser determinada sin usar información adicional procedente de los sensores onboard o del conocimiento sobre objetos presentes en la escena como marcadores artificiales. Sin embargo, para realizar una navegación autónoma, estimar este factor de escala es esencial. En el sistema TUM AR Drone se ha dado respuesta a este problema mediante una formulación estadística de máxima verosimilitud sobre las lecturas de altura absoluta del cuadricóptero (obtenidas fusionando los datos provenientes del altímetro y del barómetro) que se obtienen al hacerlo subir y bajar un metro tras el despegue (consultar el apartado 1.1.4 del anexo II para una explicación más detallada). Sin embargo, esta solución no es óptima, ya que las lecturas de altura están sujetas a error (especialmente si el suelo no es totalmente plano) y estos errores en la estimación de la escala provocan errores considerables en la estimación de la posición del cuadricóptero.

Debido a estos inconvenientes, se decidió aislar el PTAM, es decir, cortar los intercambios de información con el EKF. Al hacerlo, el filtro de Kalman extendido funcionaría exclusivamente con información procedente de los sensores onboard del cuadricóptero y de los comandos de control calculados por el PID. Se realizaron numerosas pruebas de vuelo empleando únicamente esta información pero se revelaron numerosos problemas que hacían imposible la navegación (deriva, información falsa proveniente de algunos sensores durante el despegue, etc.). Se decidió, por tanto, plantear una optimización del procesado de los datos enviados por los sensores onboard con el objetivo de poder realizar una navegación básica empleando exclusivamente esta información. Dicho procesado se presenta detalladamente en el apartado 4.2

A pesar de este proceso de optimización, existen limitaciones insalvables en la realización de una navegación precisa durante largos intervalos de tiempo empleando únicamente los sensores onboard con los que cuenta el AR Drone. La principal de ellas es la deriva que aparece en las lecturas del giroscopio en z (afectando al cálculo del yaw) y en el proceso de integración de los datos procedentes de los acelerómetros (afectando al cálculo de x e y).

Se comprende entonces la necesidad de emplear algún método que permita eliminar la deriva. Se ha optado por substituir las balizas con estimación absoluta de la posición que proporcionaba el PTAM por un sistema en base a marcadores artificiales conocido como April Tags Fiducial System, que se ha presentado anteriormente en el apartado 3.5 como detector de la posición y orientación de un tag con respecto a una cámara. Una parte de este trabajo fin de grado se va a centrar en conseguir transformar las lecturas de este sistema en un "sensor" fiable del estado del cuadricóptero mediante un procesado que se presenta detalladamente en el apartado 4.3. También es necesario comentar los aspectos relativos a como se realiza la integración y fusión de los datos procedentes de las dos diferentes fuentes mencionadas (sensores onboard y April Tags). Este aspecto se trata en el apartado 4.4.

Para llevar a cabo todas las tareas mencionadas se hizo imprescindible la realización de un estudio detallado que permita definir la implementación concreta que se realizó en el TUM AR Drone para el filtro de Kalman extendido. Dicho estudio se presenta en el apartado 4.1

4.1 Análisis de la implementación del filtro de Kalman extendido (EKF) en el TUM AR Drone

El objetivo de este apartado es describir en detalle como se ha formulado el filtro de Kalman extendido en el software TUM AR Drone. Para detalles sobre la formulación del filtro genérico, consultar el segundo apartado del anexo II.

El sistema concreto al que se va a aplicar el filtro es el cuadricóptero comercial AR Drone. Las magnitudes que se quieren analizar son la posición, orientación y velocidad del mismo, lo que determinará como se definen el vector de estado (presentado en el apartado 4.1.1), el modelo de transición de estado (apartado 4.1.2) y el modelo de observación (apartado 4.1.3).

4.1.1 Vector de estado del dron

El vector de estado del AR Drone está compuesto por las siguientes variables:

$$x(t) = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \Phi, \Theta, \Psi, \dot{\Psi})^T \in \mathbb{R}^{10} \qquad [1]$$

Dónde:

- x, y, z son las coordenadas del centro del dron (en metros) en los distintos ejes del sistema de coordenadas correspondiente al mundo real (representado en la figura 7). Dicho sistema queda fijado por como se haya colocado el dron antes de iniciar el TUM AR Drone.
- $\dot{x}, \dot{y}, \dot{z}$ son las velocidades del dron (en metros por segundo) expresadas en los distintos ejes del sistema de coordenadas correspondiente al mundo real (ver figura 7).
- Φ, Θ, Ψ son los ángulos (en grados) definidos respectivamente como roll, pitch y yaw, que representan la orientación del dron con respecto al sistema de coordenadas del mundo real (consultar figura 7). Notar que los diseñadores han establecido que los giros en los ejes x e y son positivos en sentido antihorario, mientras que el giro en el eje z lo es en sentido horario.
- $\dot{\Psi}$ es la velocidad de giro con respecto al eje z (en º/s) en grados por segundo.

4.1.2 Modelo de transición de estado

El modelo de transición de estado se define matemáticamente como $f : \mathbb{R}^n \times \mathbb{R}^d \rightarrow \mathbb{R}^n$ donde n es la dimensión del vector de estado y d es la dimensión del vector de control. Sirve para propagar el estado a través del tiempo, es decir, poder predecir si partimos de un instante de tiempo t en el que el estado es $x(\mathrm{t})$ y el comando de control activo $u(t)$ cual será el estado $x(t + \delta t)$ en el instante de tiempo $t + \delta t$.

Debido a que el dron envía precisas referencias de tiempo en microsegundos junto con la medida del sensor, podemos utilizar estos tiempos para determinar los intervalos exactos de predicción δt . La parte predictiva del filtro se lleva pues a cabo en intervalos irregulares de tiempo que están dentro del intervalo $\delta_t \approx 5 \pm 1$ ms. Para crear el modelo de predicción de estado hay que tener en cuenta tres aspectos fundamentales:

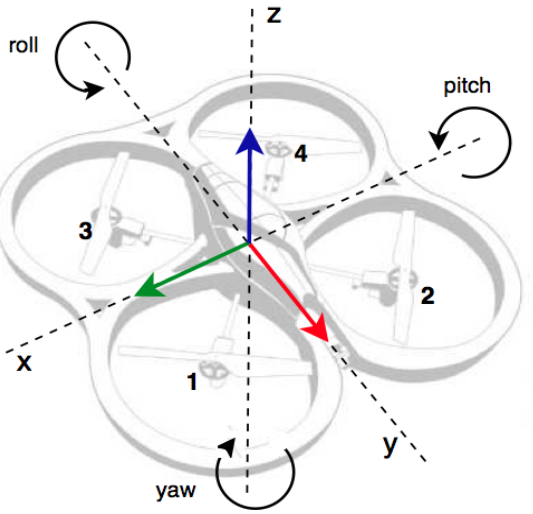


***Figura 7**. Representación del sistema de coordenadas establecido en el TUM AR Drone y el criterio de giros en cada eje (la flecha indica el sentido de giro positivo) [21].*

a) Aceleración horizontal

La velocidad horizontal del dron cambia de acuerdo con la aceleración horizontal. Esta aceleración puede ser modelizada del siguiente modo de acuerdo con la segunda ley de Newton:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \frac{1}{m} f \qquad [2]$$

Donde $m \in \mathbb{R}$ es la masa del dron, y $f \in \mathbb{R}^2$ es la suma de todas las fuerzas horizontales que actúan sobre el dron. Estas fuerzas son esencialmente dos:

1) Fuerza debida a la resistencia del aire

Corresponde a la fuerza de arrastre y la denotaremos como $f_{drag} \in \mathbb{R}^2$. Para un objeto que se mueve a una velocidad relativamente pequeña, la fuerza que actúa sobre él debido a la resistencia del aire es aproximadamente proporcional a su velocidad actual. Por tanto esta fuerza puede modelizarse como:

$$f_{drag} \propto -\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \qquad [3]$$

2) Fuerza de aceleración debida a los ángulos roll y pitch

Corresponde a la fuerza de empuje y la denotaremos como $f_{thrust} \in \mathbb{R}^2$. Se considera que los propulsores generan una fuerza constante que actúa sobre el eje z del dron. Si este está inclinado, una porción de esta fuerza actúa horizontalmente, haciendo que el dron se desplace en el plano horizontal. Esta fuerza puede modelizarse como proporcional a la proyección del eje z del dron sobre el plano horizontal:

$$f_{thrust} \propto \begin{pmatrix} \cos\Psi\ \sin\Phi\ \cos\Theta - \sin\Psi\ \sin\Theta \\ -\sin\Psi\ \sin\Phi\ \cos\Theta - \cos\Psi\ \sin\Theta \end{pmatrix} \quad [4]$$

Si se asume que las fuerzas de empuje y arrastre son constantes durante el corto periodo de tiempo considerado (δt), y juntamos la tres ecuaciones anteriores y sus constantes de proporcionalidad, la aceleración horizontal del dron vendrá dada por las siguientes ecuaciones:

$$\ddot{x}(x) = c_1(c_2(\cos\Psi\ \sin\Phi\ \cos\Theta - \sin\Psi\ \sin\Theta) - \dot{x}) \quad [5]$$

$$\ddot{y}(x) = c_1(c_2(-\sin\Psi\ \sin\Phi\ \cos\Theta - \cos\Psi\ \sin\Theta) - \dot{y}) \quad [6]$$

La constantes c_1 y c_2 son constantes del modelo: la primera define la rapidez con la que la velocidad se ajusta a un cambio de actitud (término que hace referencia al pitch y el roll que tenemos en un determinado instante), mientras que la segunda define cual es la máxima velocidad alcanzada con respecto a una determinada actitud. Se supone que el dron se comporta del mismo modo tanto en la dirección *x* como en la *y*.

b) Modelo del sistema de control

Los comandos de control tienen la siguiente forma: $u = (\bar{\Phi}, \bar{\Theta}, \bar{\dot{\Psi}}, \bar{\dot{z}}) \in [-1,1]^4$ y definen cuales son el roll y el pitch que deseamos, así como la velocidad de giro con respecto al eje z y la velocidad vertical del dron, dada como una fracción del valor máximo permitido, que es 1 o -1. Estos parámetros sirven como valores de entrada para un controlador que funciona a bordo del dron, y que ajusta las velocidades de los motores de acuerdo con los comandos recibidos. Es necesario pues establecer un modelo para este controlador, y lo que se establece es una aproximación de las velocidades de giro en el eje x (relativa al roll) y en el eje y (relativa al pitch), así como de la aceleración angular en el eje z (relativa al yaw) y la aceleración vertical en base al estado actual y a los comandos de control enviados:

$$\dot{\Phi}(x,u) = c_3(c_4\bar{\Phi} - \Phi) \quad [7]$$

$$\dot{\Theta}(x,u) = c_3(c_4\bar{\Theta} - \Theta) \quad [8]$$

$$\ddot{\Psi}(x,u) = c_5(c_6\bar{\dot{\Psi}} - \dot{\Psi}) \quad [9]$$

$$\ddot{z}(x,u) = c_7(c_8\bar{\dot{z}} - \dot{z}) \quad [10]$$

Donde las constantes de c_3 a c_8 son constantes del modelo que se determinan experimentalmente. Se asume que el comportamiento del dron es el mismo con respecto al roll y al pitch.

c) Determinación de las constantes del modelo

Las constantes del modelo que se han introducido anteriormente (de c_1 a c_8) se determinan mediante la minimización de una serie de funciones de error a partir de datos obtenidos de diversos vuelos de prueba. A continuación se explica como funciona su determinación mediante el ejemplo de las constantes c_1 y c_2. Se parte de la función:

$$E_{c_1,c_2}(c_1,c_2) = \sum_t (\dot{x}(t) - \hat{\dot{x}}(t))^2 \qquad [11]$$

Donde $\dot{x}(t)$ es la velocidad "verdadera", que se corresponde con la estimación de la velocidad en el tiempo t (ver figura 8) tras integrar todas las medidas disponibles, y $\hat{\dot{x}}(t)$ es la velocidad predicha, que se calcula como:

$$\hat{\dot{x}}(t+\delta t) = \hat{\dot{x}}(t) + \delta_t c_1 (c_2 f_{thrust,x}(t) - \hat{\dot{x}}(t)) \qquad [12]$$

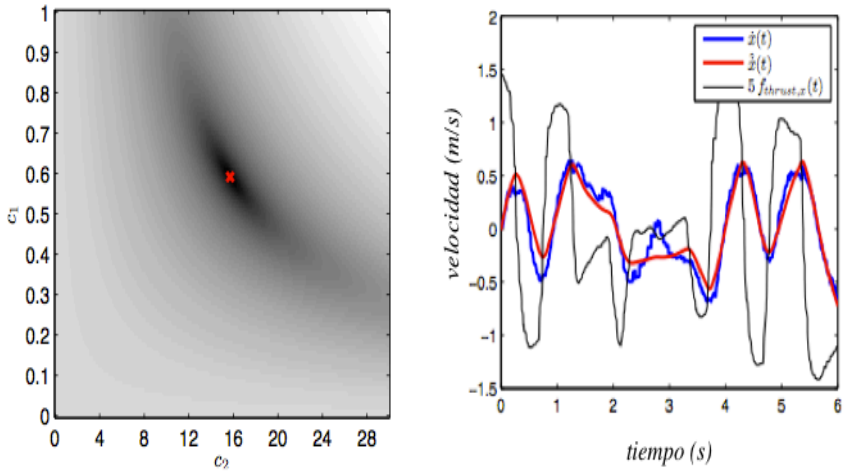


***Figura 8**. Gráficas que ilustran el método que se sigue para obtener el valor de las constantes c_1 y c_2 del modelo [21].*

Se busca que la velocidad "verdadera" ($\dot{x}(t)$, línea azul en la figura 8) y la velocidad predicha en el instante siguiente ($\hat{\dot{x}}(t)$, línea roja) difieran lo mínimo posible. La línea negra representa la fuerza de empuje en el eje x ($f_{thrust,x}(t)$). En la izquierda se muestra una imagen del proceso de minimización de E_{c_1,c_2}: la zona más oscura se corresponde con la zona de mínimo error. Se observa un claro mínimo en $c_1 = 0.6$ y $c_2 = 16$.

Una vez explicado lo anterior, puede escribirse ya la función de transición de estado completa como:

$$\begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \Phi \\ \Theta \\ \Psi \\ \dot{\Psi} \end{pmatrix}_t + \delta t \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x}(\mathrm{x}) \\ \ddot{y}(x) \\ \ddot{z}(x,u) \\ \dot{\Phi}(x,u) \\ \dot{\Theta}(x,u) \\ \dot{\Psi} \\ \ddot{\Psi}(x,u) \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \Phi \\ \Theta \\ \Psi \\ \dot{\Psi} \end{pmatrix}_{t+\delta t} \qquad [13]$$

4.1.3 Modelo de observación

El modelo de observación se define matemáticamente como $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ donde *n* es la dimensión del vector de estado y *m* la dimensión del vector observación (que representa las medidas tomadas por los sensores). Sirve para calcular las medidas estimadas que corresponderían al estado actual del dron y compararlas con el vector de medidas real, denotado por $z_k \in \mathbb{R}^m$. Debido a que tenemos dos fuentes distintas y asíncronas de observación es necesario considerar dos modelos:

a) Modelo de observación para el PTAM

El sistema de SLAM genera una estimación de la posición del dron para cada frame de video, y dicha información se trata como una observación directa de los parámetros de estado, es decir:

$$h_{PTAM}(x) = (x, y, z, \Phi, \Theta, \Psi)^T \in \mathbb{R}^6 \qquad [14]$$

El vector de medidas se trata de la siguiente forma:

$$z_{PTAM} = \log(E_{DC} E_C) \qquad [15]$$

Donde $E_C \in SE(3)$ representa la posición de la cámara que estima el PTAM, $E_{DC} \in SE(3)$ representa la transformación que convierte el sistema de coordenadas de la cámara al sistema de coordenadas del dron, y $\log : SE(3) \rightarrow \mathbb{R}^6$ es la transformación que convierte un elemento del grupo euclídeo SE(3) a la representación $(x, y, z, \Phi, \Theta, \Psi)$.

b) Modelo de observación para los sensores onboard

El modelo que transforma las variables de estado a la observación equivalente tiene la siguiente forma:

El modelo que transforma las variables de estado a la observación equivalente tiene la siguiente forma:

$$h_{Onboard}(x) = \begin{pmatrix} \cos(\Psi)\,\dot{x} - \sin(\Psi)\,\dot{y} \\ \sin(\Psi)\,\dot{x} + \cos(\Psi)\,\dot{y} \\ z \\ \Phi \\ \Theta \\ \Psi \end{pmatrix} \in \mathbb{R}^6 \qquad [16]$$

El vector de medidas reales tiene la siguiente forma:

$$z_{Onboard} = \begin{pmatrix} \dot{x}_{proc} \\ \dot{y}_{proc} \\ z_{proc} \\ \Phi_{proc} \\ \Theta_{proc} \\ \Psi_{proc} \end{pmatrix} \in \mathbb{R}^6 \qquad [17]$$

Donde el subíndice *proc* hace referencia al procesado, que constituye un proceso complejo que se detalla en profundidad en el apartado 4.2.

4.2 Procesado de los datos procedentes de los sensores onboard.

Como se ha comentado en el apartado precedente, tras aislar el PTAM debido a las limitaciones que suponía en la navegación, se realizaron pruebas de vuelo empleando exclusivamente los sensores onboard del cuadricóptero y los comandos de control calculados por el PID, y se observó que no era viable navegar si se empleaba el planteamiento original del TUM AR Drone. En esta sección se expone el análisis realizado de la información sobre la posición y orientación del dron que proporcionan

los sensores onboard y el "post-procesado" que se ha desarrollado para mejorar las lecturas antes de ser integradas en el filtro de Kalman como observaciones. Se han dividido estas señales en 4 apartados: velocidad horizontal, altura, ángulos roll y pitch y ángulo yaw. En cada uno de ellos se explicará en primer lugar como se calcula la observación y en segundo lugar como se han optimizado estas.

En la figura 10 se han desarrollado las distintas etapas del procesado de las lecturas de los sensores onboard del cuadricóptero. En primer lugar se encuentra el preprocesado que lleva a cabo el propio firmware del AR Drone. A continuación se envían las lecturas al ordenador a través de una red LAN inalámbrica y son convertidas a formato ROS por el driver del AR Drone, que ya se explicó en el apartado 3.1. A continuación se detalla el procesado de las medidas en el ordenador, compuesto de tres etapas: en primer lugar el cálculo de la observación, es decir, la conversión de los datos que provienen del AR Drone en una observación de variable de estado del filtro. Para los sensores onboard estás variables son $(\dot{x},\dot{y},z,\Phi,\Theta,\Psi)^T$ tal y como se ha comentado en el apartado 4.1.3. En segundo lugar está el filtrado de los datos y en tercero la calibración de las medidas. Es importante destacar que no todas las lecturas pasan por los tres procesos. Las observaciones finales se corresponden con el vector de la ecuación 17.

4.2.1 Velocidad horizontal

Se denotan como v_x y v_y a las velocidades en los ejes *x* e *y* del sistema de referencia del cuadricóptero, es decir, las velocidades frontal y lateral del mismo. Esta información es enviada por el AR Drone con una frecuencia de 200 Hz (5 ms) a través de una wireless LAN y es interpretada por el driver implementado en ROS (que ya se ha descrito anteriormente en el apartado 3.1). Las citadas velocidades son estimadas por el firmware del cuadricóptero, que tiene implementado un filtro donde se fusionan los datos provenientes de dos fuentes: por un lado están los acelerómetros de la IMU cuyas lecturas se envían a un integrador del propio firmware del AR Drone para dar unos datos de velocidad horizontal. Por otro lado está la cámara inferior (320 x 240 pixeles) del cuadricóptero que envía los frames de imagen que captura a un módulo del firmware que aplica una serie de algoritmos de flujo óptico mediante los que se obtiene una lectura de las velocidades lineales en *x* e *y*. Todo este proceso aparece reflejado en la figura 10.

4.2.1.1 Cálculo de la observación equivalente

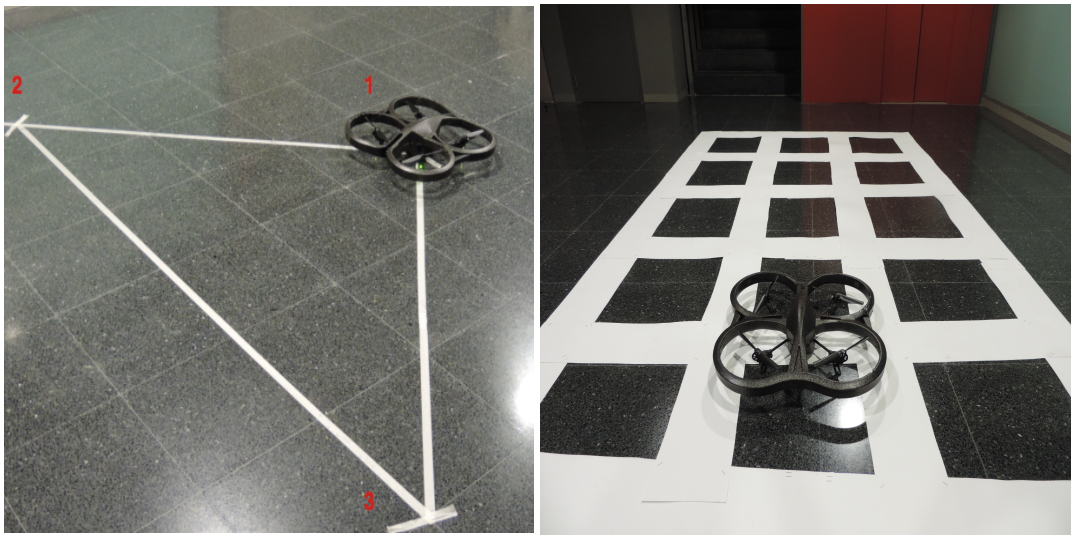
Para ello simplemente hay que proyectar las v_x e v_y según el yaw actual del cuadricóptero, para que dichas velocidades queden expresadas en el sistema de referencia del dron. También se hace la división entre mil para pasar a *m/s* la información que publica el driver (que está en *mm/s)*:

$$v_{x_{proc}} = \frac{\sin\Psi\ v_x + \cos\Psi\ v_y}{1000} \quad [18]$$

$$v_{y_{proc}} = \frac{\cos\Psi\ v_x - \sin\Psi\ v_y}{1000} \quad [19]$$

4.2.1.2 Optimización de las observaciones

Se ha planteado un experimento para estudiar la influencia de distintos factores ambientales en la estimación de la velocidad horizontal que realiza el firmware del AR Drone. Dicho experimento consiste en ordenar al AR Drone que describa una trayectoria triangular, cuya proyección se ha marcado sobre el plano horizontal del suelo (figura 10.a) con el fin de poder realizar la estimación visual de la desviación de la misma. Los vértices de esta trayectoria constituyen los *targets* o puntos objetivo a los que se ha de dirigir el dron, y se corresponden con $(x,y,z,\Psi) = (1.2,1.2,1,0)$ para el punto 1, $(x,y,z,\Psi) = (-1.2,1.2,1,0)$ para el punto 2 y $(x,y,z,\Psi) = (0,0,1,0)$ para el punto 3. La maniobra se llevará a cabo secuencialmente para poder facilitar la comparación con el patrón que debería seguir. Esto quiere decir que se le ordenará al dron que aterrice cada vez que alcance un punto objetivo para luego volver a despegar y dirigirse al siguiente *target*, y así hasta completar la maniobra.



(a) *(b)*

Figura 9. *Comparación entre un suelo tipo 0 (a) y un suelo tipo 1 (b). En (a) también se observa la trayectoria exacta que debería de seguir el AR Drone según las especificaciones externas introducidas por el usuario en el controlador PID.*

La primera de las pruebas que se va a llevar a cabo tiene lugar sobre un tipo de suelo al que nos referiremos como tipo 0 (figura 9.a), el suelo existente en el laboratorio donde se realizaron los experimentos, caracterizado por tener una textura homogénea y color oscuro.

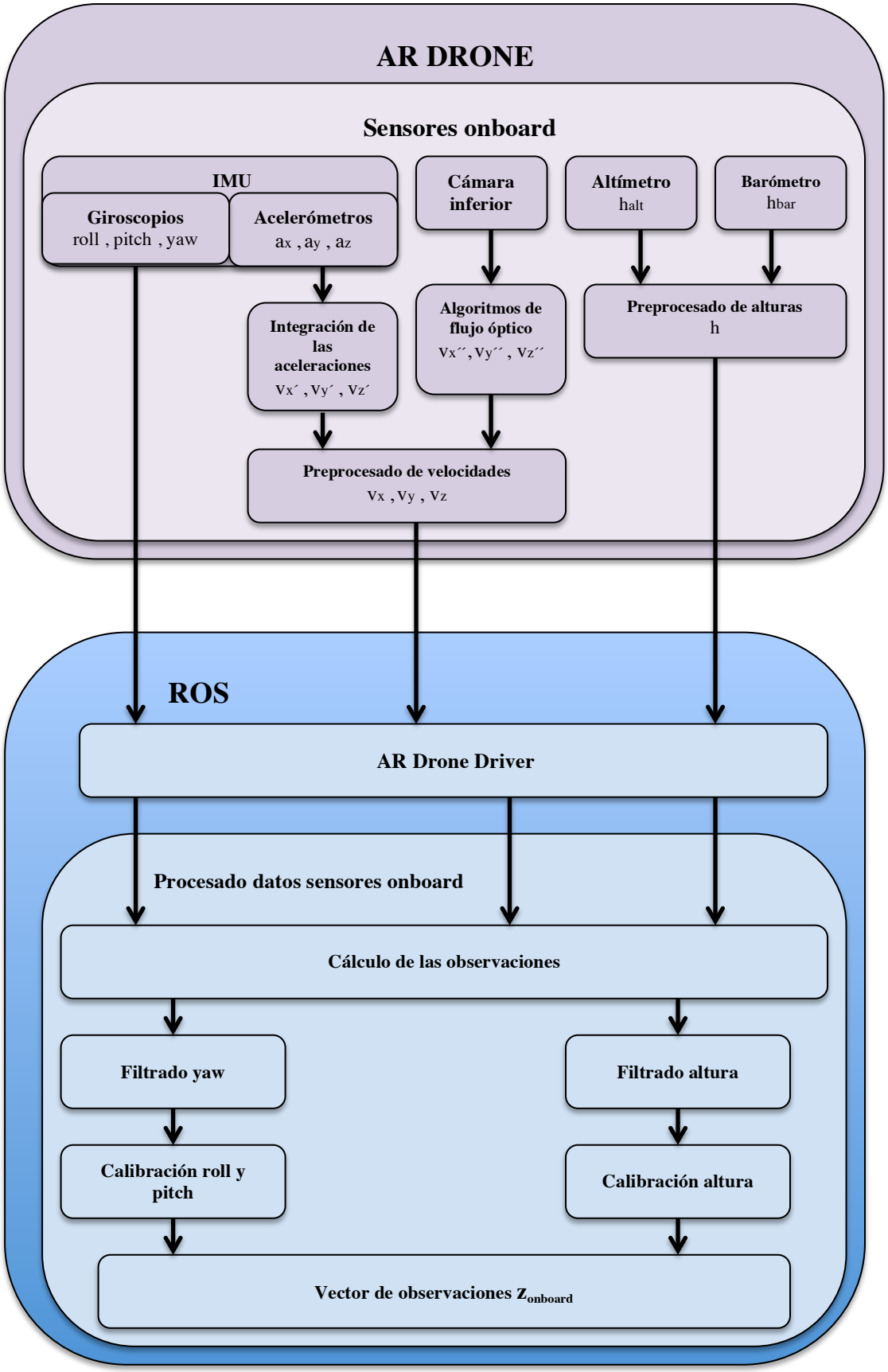


Figura 10. *Esquema detallado del procesado que se aplica a los datos procedentes de los sensores onboard del AR Drone*

En la figuras 11, 12 y 13 puede verse cual es el comportamiento del AR Drone cuando se le pide realizar la maniobra. En la tabla 1 se muestran los resultados del experimento, constatándose que existe un considerable error de posicionamiento.

Punto	Posición objetivo en x [m]	Posición real en x [m]	Posición objetivo en y [m]	Posición real en y [m]
1	1.2	1.602	1.2	1.602
2	-1.2	-2.406	1.2	2.004
3	0	-2.808	0	-2.010

Tabla 1. *Resumen de los resultados del experimento en un suelo tipo 0.*

A continuación se planteó la realización de la misma maniobra en lo que denominaremos un suelo de tipo 1, caracterizado por tener una textura heterogénea, con franjas de color oscuro contrastadas con franjas de color claro. Debido a que como ya se ha comentado, el suelo del laboratorio es de tipo 0, hubo que ingeniar un sistema que permitiese simular un suelo de tipo 1.

Para ello, se construyó un entramado con cartulinas blancas tal y como se muestra en la figura 9.b. Obviamente esto no es necesario si la habitación dispusiese de un suelo tipo 1 más común como por ejemplo un suelo de madera, baldosas de colores vivos, etc. En la figuras 14, 15 y 16 puede verse cual es el comportamiento del AR Drone cuando se le pide realizar la maniobra. En la tabla 2 se muestran los resultados del experimento, constatándose que el posicionamiento del AR Drone es mucho más preciso que el que se observó en el suelo de tipo 0.

Punto	Posición objetivo en x [m]	Posición real en x [m]	Posición objetivo en y [m]	Posición real en y [m]
1	1.2	1.2	1.2	1.2
2	-1.2	-1.803	1.2	1.401
3	0	-0.201	0	-0.402

Tabla 2*. Resumen de los resultados del experimento en un suelo tipo 1.*

Al ver que existían estas diferencias según el tipo de suelo, se decidió plantear un último experimento que permita comparar las velocidades que devuelven los sensores onboard en dos escenarios diferentes con las velocidades 'pseudoreales' para estudiar las discrepancias entre ambas y el efecto de las condiciones ambientales.

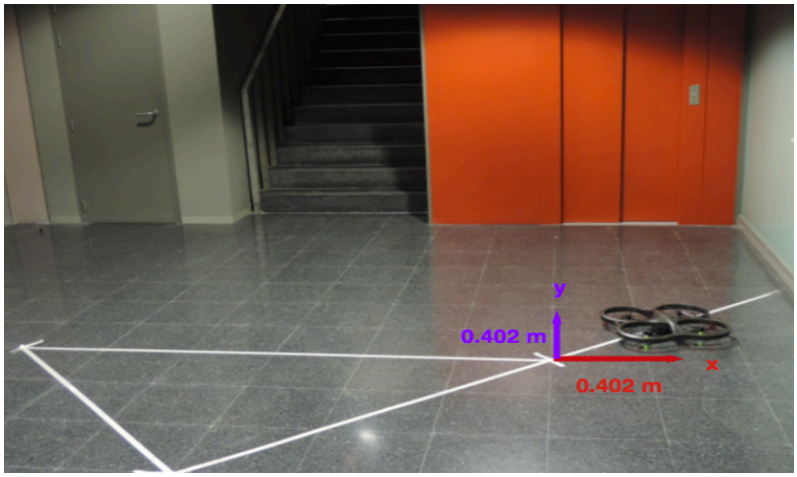


Figura 11. *Posición real del AR Drone en la que se considera que ha alcanzado el punto 2 en un suelo tipo 0. Se observa un error absoluto de en torno a 0.402 m tanto en x como en y.*

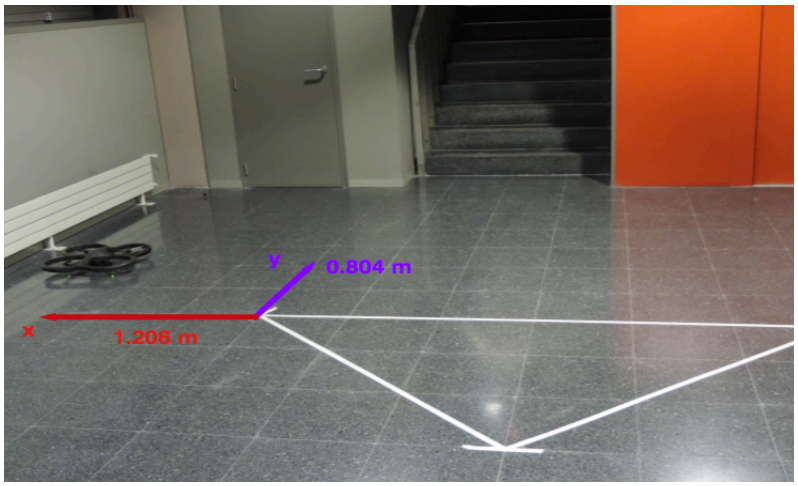


Figura 12. *Posición real del AR Drone en la que se considera que ha alcanzado el punto 3 en un suelo tipo 0. Se observa un error absoluto de en torno a 1,206 m en x y 0.804 en y.*

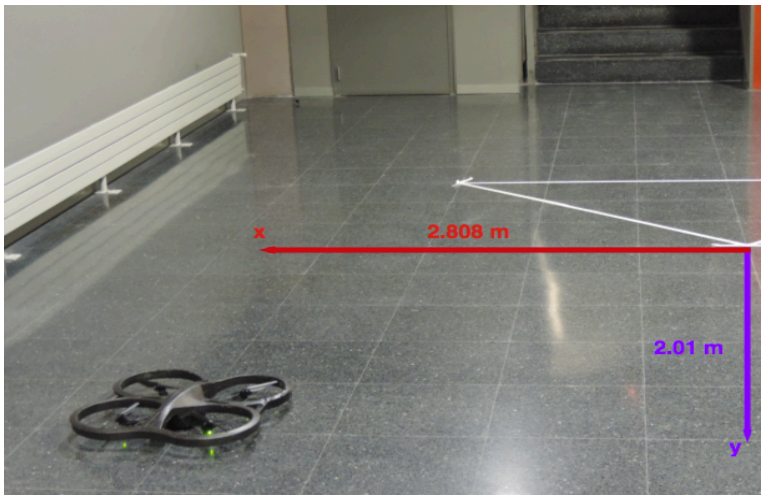


Figura 13. *Posición real del AR Drone en la que se considera que ha alcanzado el punto 1 en un suelo tipo 0. Se observa un error absoluto de en torno a 2.808 m en x y 2.01 m en y.*

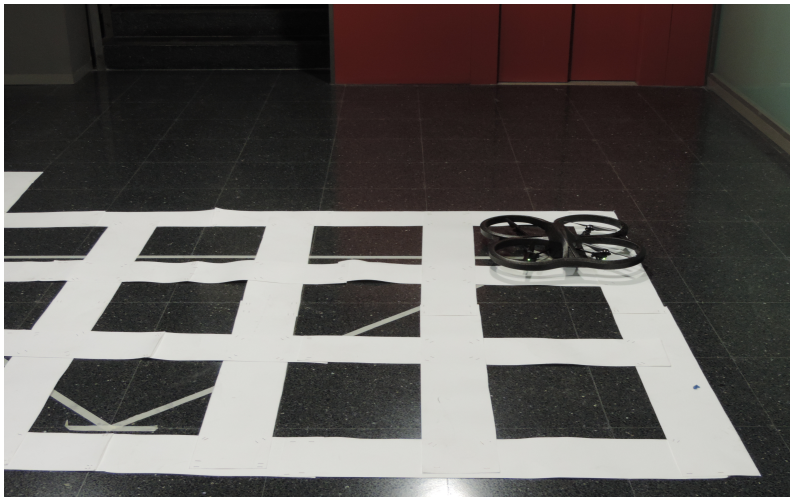


Figura 14. *Posición real del AR Drone en la que se considera que ha alcanzado el 2 en un suelo tipo 1. Se observa que el error es prácticamente nulo.*

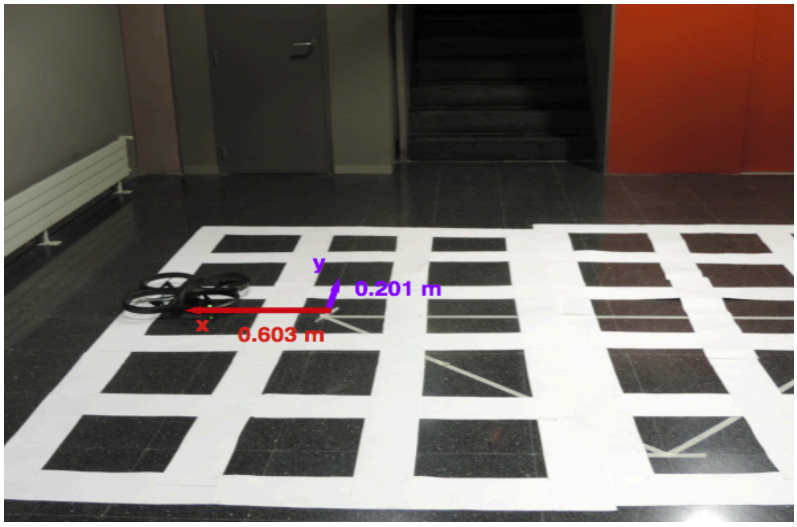


Figura 15. *Posición real del AR Drone en la que se considera que ha alcanzado el 2 en un suelo tipo 1. Se observa que el error absoluto de 0.603 m en x y 0.201 en y.*

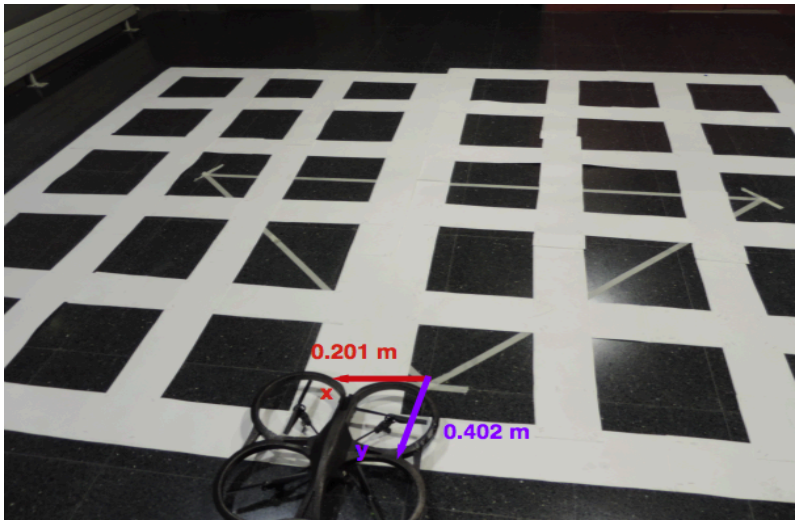


Figura 16. *Posición real del AR Drone en la que se considera que ha alcanzado el 2 en un suelo tipo 1. Se observa un error absoluto de 0.201 m en x y 0.402 en y.*

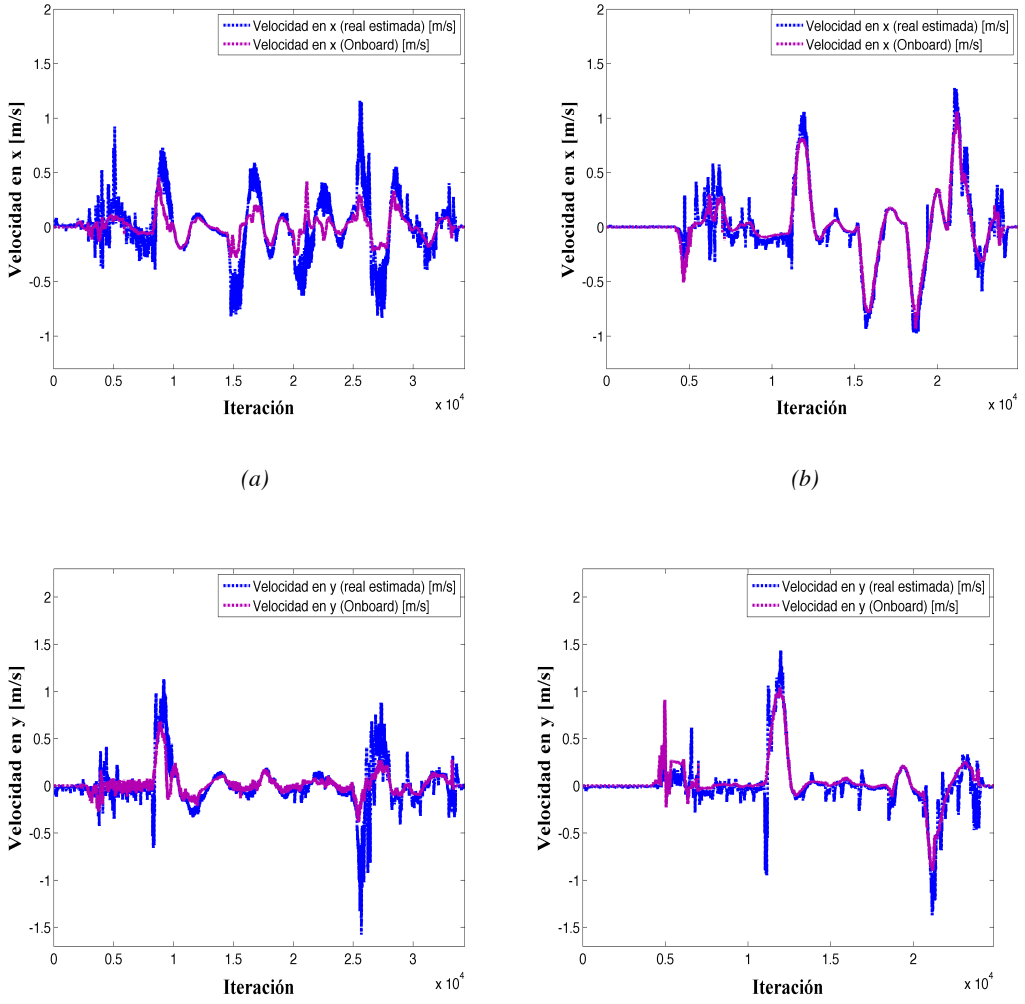


Figura 17. *(a) Representación de la velocidad real estimada en x (color azul) en comparación con la velocidad en x calculada por el firmware del AR Drone (color morado) en un suelo tipo 0. (b) Representación de las mismas magnitudes para un suelo de tipo 1. (c) Representación la velocidad real estimada en y (color azul) en comparación con la velocidad en y calculada por el firmware del AR Drone (color morado) en un suelo tipo 0. (d) Representación de las mismas magnitudes para un suelo de tipo 1.*

El experimento fue llevado a cabo una vez completado y calibrado el sistema de navegación final (que incluye balizas artificiales) pero se ha creído conveniente incluirlas en este apartado, ya que proporcionan una estimación precisa de la posición con la que obtener una estimación altamente fiable de la velocidad real del AR Drone.

Los resultados del experimento se muestran en la figura 17. En la gráfica (a) se representa la velocidad real estimada en *x* (color azul) en comparación con la velocidad en *x* calculada por el firmware del AR Drone (color morado) en un suelo tipo 0. En la gráfica (b) que representan las mismas magnitudes para un suelo de tipo 1. La gráfica (c) representa la velocidad real estimada en *y* (color azul) en comparación con la velocidad en *y* calculada por el firmware del AR Drone (color morado) en un suelo tipo 0. La gráfica (d) representa las mismas magnitudes para un suelo de tipo 1. Las gráficas

muestran que las diferencias entre la velocidad real estimada y la que devuelven los sensores onboard se reducen notablemente en un suelo tipo 1. Esto es especialmente notorio en los tramos en los que la velocidad es alta, en los que el firmware del AR Drone tiende a subestimar la velocidad de manera más acusada en el suelo tipo 0. Este hecho provocará que, en virtud del proceso de integración de las velocidades, el cuadricóptero se desplace durante más tiempo en un suelo tipo 0 (tiempo total de maniobra de 3,5 segundos) que en uno de tipo 1 (tiempo total de maniobra de 2.5 segundos) para alcanzar una misma posición. Esto es lo que provoca que en los suelos tipo 0 el AR Drone se pase una distancia mayor con respecto al punto objetivo (ver figuras 11, 12 y 13) que en los de tipo 1 (ver figuras las 14, 15 y 16).

4.2.2 Altura

4.2.2.1 Cálculo de la observación

La altura del dron no se puede tratar como una observación directa, ya que las desviaciones originadas por un suelo no plano o una estimación imprecisa de la escala provocaría valores inestables y oscilantes de la altura estimada del cuadricóptero y consecuentemente de su velocidad vertical. Es por este motivo que se calcula la altura relativa *h* del dron. Su valor se computa cada 40 ms, valor determinado por la frecuencia con la que el AR Drone envía las lecturas de la altura absoluta del mismo. Dicha información se obtiene de preprocesar en un filtro presente en el firmware la información procedente del altímetro y del barómetro. Todo este proceso aparece reflejado en la figura 10.

Previo al uso de la señal de altura procedente de los sensores onboard, al igual que en el procedimiento que ya se propone en la implementación del TUM AR Drone, se realiza un postprocesado que consiste en descartar observaciones que impliquen variaciones de altura entre medidas superiores a 15 cm en 40 ms (lo que corresponde a una velocidad vertical de unos 3.75 m/s), ya que se asume que ha aparecido una discontinuidad en el suelo. Con el objeto de eliminar estos saltos, el cálculo de la altura instantánea se realiza del siguiente modo:

$$z_{proc} = z(t-\delta t) + h(t) - h(t-\delta t) \qquad [20]$$

Donde:

- $z(t-\delta t)$ es la altura del dron proporcionada por el filtro tras la última observación de altura.

- $h(t) = h_{nav} - baseline_{Onboard}$ es la altura relativa medida en la iteración actual. Esta altura relativa se computa como la diferencia entre la lectura del valor absoluto

de la altura del dron enviada por el firmware del mismo (que se denotará como h_{nav}) y una línea de base (que denotamos como $baseline_{Onboard}$). El criterio de detección de saltos establece que $h(t) < 0.150$. Cuando el valor de $h(t)$ supera este umbral (ya sea por una discontinuidad de altura o por una elevación del dron) se actualiza $baseline_{Onboard}$ al valor del h_{nav} en ese instante.

- $h(t-\delta t)$ es la altura relativa medida en la iteración anterior, es decir, hace 40 ms.

4.2.2.2 Optimización de las observaciones

En este apartado se describirán los nuevos métodos de postprocesado que se han implementado para mejorar las observaciones de la altura del cuadricóptero a nivel de precisión, mediante un proceso de postfiltrado que se presenta en el apartado a), y de exactitud, mediante un proceso de calibración que se presenta en el apartado b).

a) Filtrado.

Se ha llevado a cabo un estudio comparativo de las lecturas de altura proporcionadas por los sensores onboard frente a las observaciones procesadas que se introducen al filtro para analizar la respuesta en despegue y vuelo.

Para lograrlo se planteó el siguiente experimento: se estableció mediante el piloto automático una maniobra simple de despegue, en la que el cuadricóptero se eleva desde el suelo (altura 0 m) hasta una cota de 1 m de altura, manteniéndose en ella durante unos segundos. Los datos de altura obtenidos se muestran en la figura 18, en la que se representan las lecturas de altura calculadas por firmware AR Drone (en color morado) y las observaciones procesadas según el planteamiento inicial del TUM AR Drone introducidas al filtro (en azul) frente al tiempo.

Tal y como se aprecia en la gráfica, después de la maniobra de despegue la altura estimada tras el postprocesado que se ha comentado en el apartado 4.2.2.1 se sitúa en torno a los -1 metros (lo cual es imposible hablando en términos de altura absoluta) generando un error de 2 metros con respecto a la altura real que está siendo medida por los sensores onboard. La navegación con el uso directo de estas medidas solo podría ser aceptable si la varianza de las señales de balizas externas, como los marcadores naturales que empleaba el PTAM, fuese muy inferior a la de los sensores onboard a la hora de introducir las observaciones en el filtro de Kalman, lo que significaría que las primeras tendrían un peso mucho mayor en el cálculo de la observación. Sin embargo, en este trabajo se ha buscado optimizar al máximo el funcionamiento de todos los sensores del cuadricóptero, por lo que hubo buscar una solución a este problema.

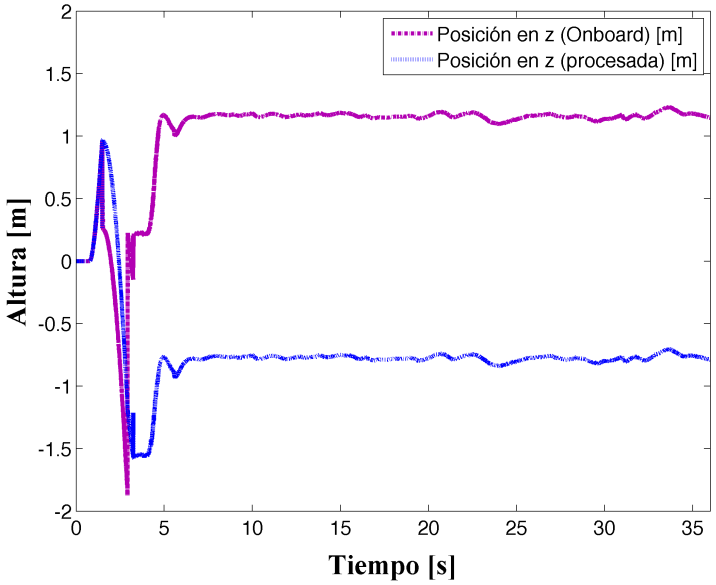


Figura 18. *Representación gráfica de las lecturas de altura calculadas por el firmware del AR Drone (en color morado) y las observaciones procesadas según el planteamiento inicial del TUM AR Drone (en azul) frente al tiempo.*

Si se observan los primeros 5 segundos de la gráfica 18, se constata que se produce una oscilación muy fuerte que pasa de valores de en torno a 1.5 m a -0.3 m en unos pocos segundos. Esto, obviamente, es imposible y no se corresponde con lo que ocurre en la realidad, por lo que hay que concluir que se trata de un comportamiento anómalo de los sensores con el que hay que contar siempre. La utilización de un cálculo acumulativo de la altura como el que se ha planteado en el apartado 4.2.2.1 tiene la ventaja de que permite detectar discontinuidades en la elevación del suelo, pero la desventaja de que estos datos erróneos al comienzo pueden estropear todo el planteamiento. Se puede entender que si tomamos un valor de $z(t-\delta t)<0$ y luego se comienza a sumar la altura relativa, vamos a obtener un valor mucho menor del que debería de obtenerse en realidad: por ejemplo si resulta que $z(t-\delta t)=-0.3$ en lugar de 0, y que la altura relativa correctamente calculada es $h(t)-h(t-\delta t)=0.5$, el valor final que se introduce en el filtro será de $z_{proc}=0.2$, cuando en realidad debería de ser 0.5.

El planteamiento que se ha seguido para la corrección del transitorio en el despegue está basado en un contador interno. Empleando una utilidad de ROS que nos permite medir el tiempo del sistema, se guarda como referencia de tiempos el primer paquete de altura enviado por el AR Drone que contiene una altura distinta de cero (lo que implica el inicio del despegue) y a partir de ahí se esperan 4.5 segundos antes de poner a funcionar el cálculo de alturas. Este valor se ha determinado empíricamente tras la realización de un gran número de experimentos, comprobándose que los valores enviados por el firmware del AR Drone tras el transitorio (valores correctos de altura) coinciden exactamente con los valores introducidos en el filtro.

Con el fin de examinar el nuevo sistema se han realizado una serie de pruebas para comprobar la precisión que tenemos en la estimación de la altura. Estas consisten en despegar el cuadricóptero y llevarlo "manualmente", es decir, por medio de una aplicación que nos permite tomar el control del cuadricóptero a través del teclado de nuestro ordenador, a una serie de alturas (0.9 m, 1.2 m, 1.5 m y 1.8 m) que se han medido y marcado en la pared del laboratorio tal y como se muestra en la figura 19.

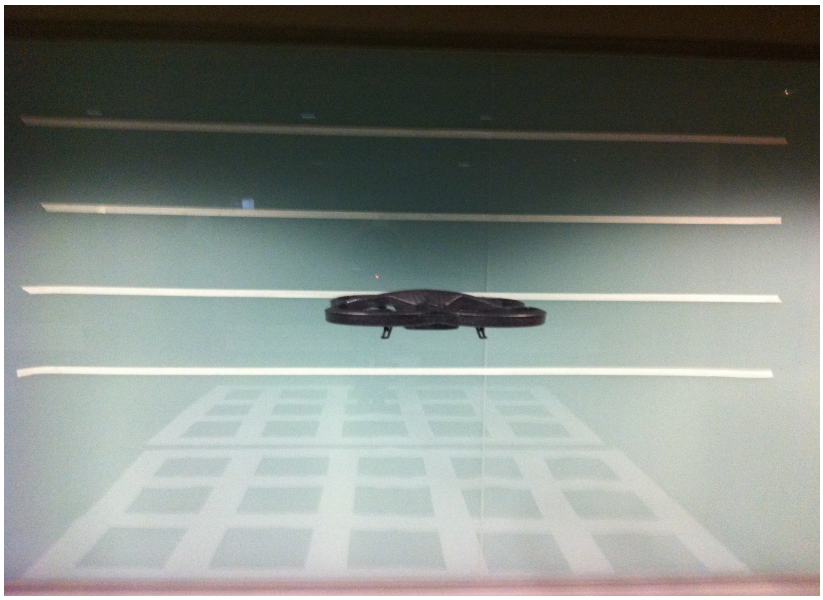


Figura 19. *Fotografía que ilustra el montaje del experimento para estudiar el funcionamiento de los sensores de altura. Las líneas blancas son referencias de alturas: 0.9 m, 1.2 m, 1.5 m y 1.8 m (de abajo a arriba).*

Las gráficas obtenidas se muestran en la figura 20. En ella se representan las lecturas de altura de los sensores onboard del AR Drone en (color azul) y de las observaciones de altura introducidas al filtro tras pasar por el nuevo procesado implementado (color morado) frente al tiempo para una altura de 90 cm (a), 1.2 m (b), 1.5 m (c) y 1.8 m (d).

Se observa claramente como el nuevo filtro introducido elimina el transitorio del despegue del cuadricóptero, y que posteriormente las lecturas de altura del AR Drone coinciden exactamente con las observaciones que se introducen en el filtro. La única diferencia se observa cuando se produce aterrizaje, ya que el firmware del AR Drone siempre envía una lectura de altura igual a cero cuando el AR Drone ha completado el aterrizaje, mientras que las observaciones se quedan en un valor de en torno 0.15 por encima. Esto se debe a la tolerancia que se establece para cambiar los baselines que se ha comentado en el apartado 4.2.2.1.

En la tabla 3 se presenta un resumen de los datos representados en la figura 20 analizados estadísticamente. Se constata que el valor medio para cada altura está muy próximo a la referencia real a la que se volaba el cuadricóptero en cada caso.

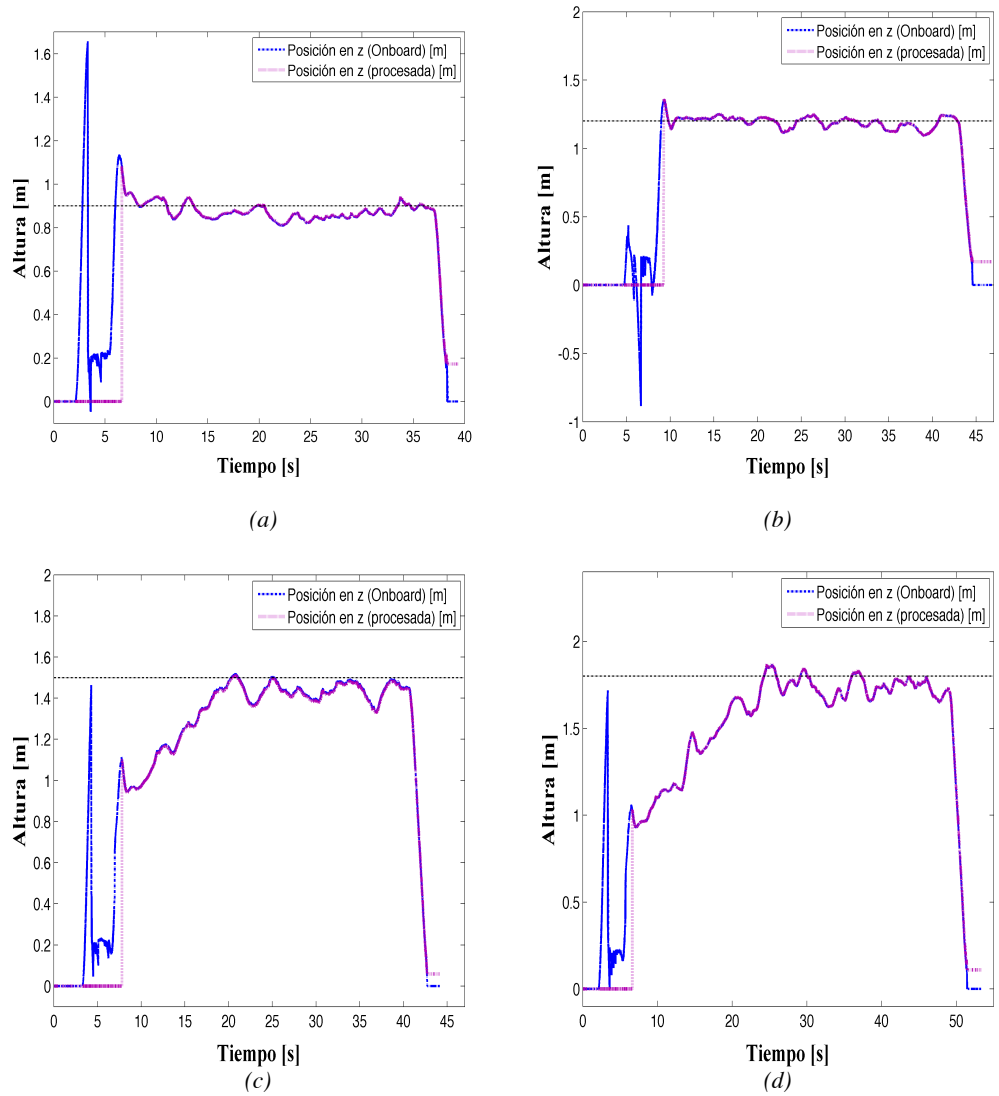


***Figura 20**. Representación gráfica de las lecturas de altura de los sensores onboard del AR Drone en (color azul) y de las observaciones de altura introducidas al filtro tras pasar por el nuevo procesado implementado (color morado) frente al tiempo para una altura de 90 cm (a), 1.2 m (b), 1.5 m (c) y 1.8 m (d).*

Valor real [m]	Valor medio estimado [m]	Desviación típica [m]
0	0	0
0.9	0,8776	0,0308
1.2	1,1857	0,0396
1.5	1,4413	0,0382
1.8	1,7505	0,0564

***Tabla 3.** Resumen del análisis estadístico aplicado sobre los datos representados en las gráficas de la figura 21.*

También se observa que cuanto mayor es la altura mayor desviación presentan los datos y mayor error absoluto existe entre la estimación y el valor real. Por último hay que destacar que parte de las oscilaciones que se observan en la gráfica se corresponden con

oscilaciones reales de la altura del dron con respecto a la cota de vuelo que se pretende mantener, generadas por los comandos de estabilización que tenemos que enviar a través del teclado para que el dron se mantenga exactamente en la cota deseada.

b) Calibración

Una vez que se han estimado los valores medios para cada posición se va a llevar a cabo una calibración las observaciones para evitar los errores sistemáticos. Se han estudiado diversos modelos de calibración que se resumen en la tabla 4 junto con sus coeficientes de determinación. De entre todos ellos, se decidió escoger el modelo de ajuste lineal debido a que presenta un coeficiente de determinación elevado que apenas se ve mejorado con otros modelos más complejos de tipo polinómico.

Tipo de modelo	Ecuación	R^2
Lineal	$z_{real} = 1.031\ z_{estim} - 0.0036$	0.99964
Polinómico de grado 2	$z_{real} = 0.011\ z_{estim}^2 + 1.0125\ z_{estim} - 0.0003$	0.99968
Polinómico de grado 3	$z_{real} = -0.0283\ z_{estim}^3 + 0.0892\ z_{estim}^2 + 0.9607\ z_{estim} + 0.0004$	0.99969
Polinómico de grado 4	$z_{real} = -0.6259\ z_{estim}^4 + 2.47\ z_{estim}^3 - 3.1268\ z_{estim}^2 + 2.2903\ z_{estim} - 5e\text{-}11$	1

Tabla 4. *Resumen de la formulación matemática de los modelos estudiados así como sus coeficientes de determinación para la altura.*

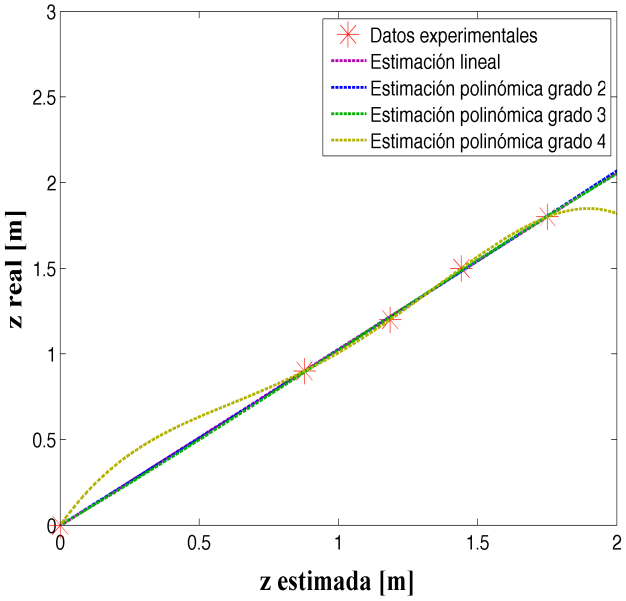


Figura 21. *Representación gráfica de cómo ajustan los diferentes modelos de calibración. Los datos experimentales se representan mediante un asterisco de color rojo, la estimación lineal media trazo discontinuo de color morado, y las estimaciones polinómicas de grado 2, 3 y 4 de color azul, verde y amarillo respectivamente.*

Este hecho se ve claramente en la figura 21 donde apenas se aprecia la diferencia entre el modelo lineal y el polinómico de grado 2 y grado 3. El ajuste del modelo polinómico de grado 4 muestra un caso evidente de regresión sobreajustada en la que a pesar de que el coeficiente de regresión es muy alto el polinomio no es capaz de generalizar el comportamiento del sensor.

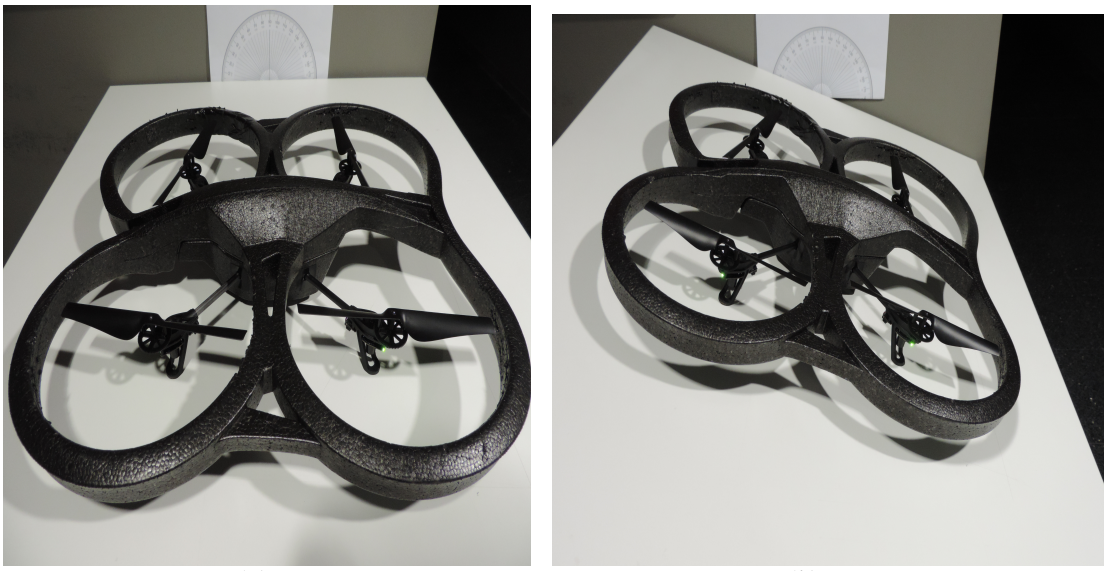
4.2.3 Roll y pitch

4.2.3.1 Cálculo de la observación

Los valores del roll y el pitch Φ,Θ pueden ser tratados como observaciones directas de las respectivas variables de estado ya que los sensores que los miden están libres de deriva y tienen una precisión elevada. La información sobre los ángulo de giro en x e y es calculada por el giroscopios correspondientes de la IMU del AR Drone y es enviada con una frecuencia de 200 Hz (5 ms) a través de una wireless LAN siendo interpretada por el driver implementado en ROS que ya se ha descrito anteriormente. Todo este proceso aparece reflejado en la figura 10.

4.2.3.2 Optimización de las observaciones.

Se decidió llevar a cabo un estudio de los datos proporcionados por estos sensores para evaluar si existía un margen de mejora en su procesado. Se diseñó un experimento basado en el empleo de una mesa de orientación regulable, midiéndose los diferentes ángulos generados por la misma mediante un transportador pegado a la pared del laboratorio. Inicialmente se orienta la mesa en línea con el cero del transportador (ver figura 22.a) y se hace coincidir el centro de este con el de la mesa.



(a) *(b)*

Figura 22. *Fotografías que muestra el montaje empleado para medir el roll y el pitch del cuadricóptero. La imagen (a) se corresponde con una medida de pitch de 0º, mientras que la (b) a una de 10º.*

Para medir el pitch en concordancia con el sistema de coordenadas definido en la figura 7, lo que hacemos es orientar la cámara del AR Drone en paralelo al plano de la pared, y giramos la mesa en sentido horario (como se muestra en la figura 22.b) para medir ángulos negativos y en sentido antihorario para los ángulos positivos. Para medir el roll, lo que hacemos es girar el AR Drone un ángulo de 90º en sentido horario con respecto al plano de la mesa. De este modo, cuando se gire la misma en sentido horario se obtendrán ángulos de roll negativos, y cuando se haga en sentido antihorario se obtendrán ángulos positivos. Los resultados de los experimentos se muestran en las figura 23.

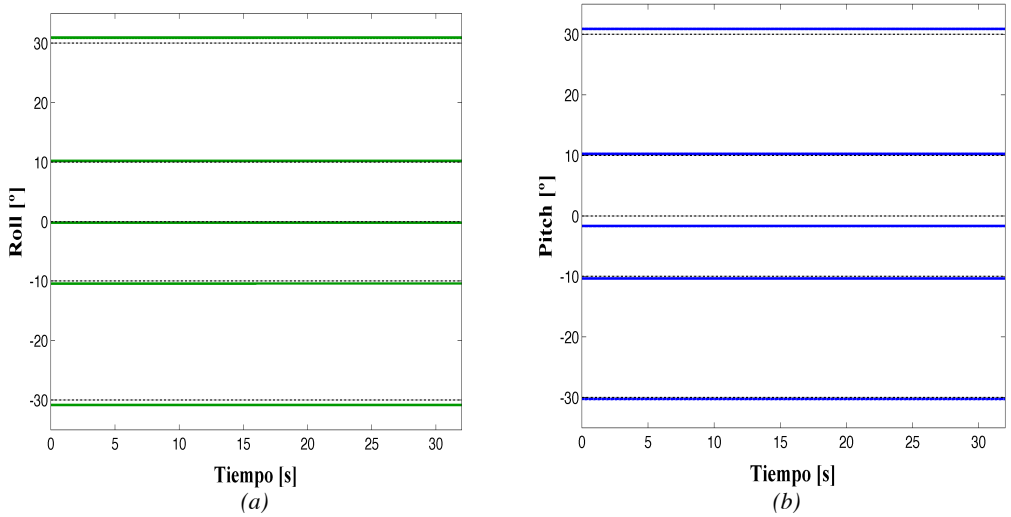


***Figura 23**. Lecturas de estado para unos ángulo reales de -30º, -10º, 0º, 10º, 30º para el pitch (a) y para el roll (b).*

Valor real [º]	Valor medio estimado [º]	Desviación típica [º]
-30	-30,5764	0,39188
-10	-10,3406	0,02275
0	-1,60762	0,02040
10	10,2585	0,06927
30	30,9511	0,06333

(a)

Valor real [º]	Valor medio estimado[º]	Desviación típica [º]
-30	-30,9587	0,08054
-10	-10,3606	0,03642
0	-0,26355	0,05463
10	10,2835	0,03671
30	30,9276	0,05573

(b)

***Tabla 5.** Resumen del análisis estadístico sobre los datos obtenidos para el pitch (a) y para el roll (b).*

También se ha realizado un análisis estadístico de los datos cuyos resultados se muestran en la tabla 5. Debido a que dichos resultados no se correspondían exactamente con el valor real se han estudiado diversos modelos de calibración con el objetivo de

afinar al máximo la precisión de la observación que se introducirá al filtro. Tanto para el pitch como para el roll se valoraron diversos modelos de calibración tal y como se muestran en la tabla 6 junto con sus respectivos coeficientes de determinación. Se decidió escoger un ajuste mediante un polinomio de grado 4 para ambas magnitudes, ya que a pesar de que el coeficiente de determinación es muy alto incluso con un modelo lineal, era el ajuste mediante este polinomio el que reducía al máximo el error en el origen (ver figura 24). Este punto es el que más interesa calibrar por el hecho de que tanto el roll como el pitch se mantienen la mayor parte del tiempo ene este valor, salvo en maniobras de vuelo bruscas donde el dron llega a inclinarse un cierto ángulo, pero nunca fuera del rango ±10º. El modelo escogido también funciona bien en ese rango de datos sin sobreajustar.

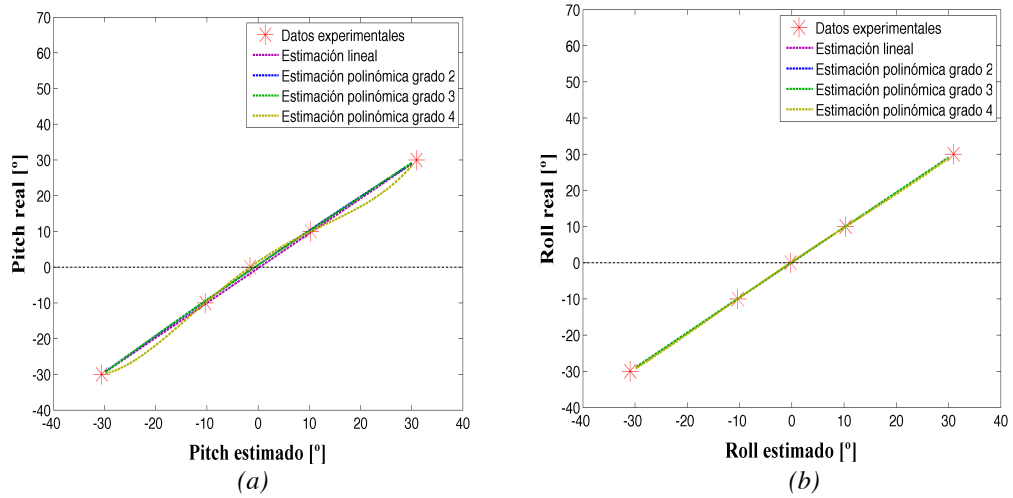
Tipo de modelo	Ecuación	R^2
Lineal	$pitch_{real} = 0.9737\ pitch_{est} - 0.2561$	0.9989
Polinómico de grado 2	$pitch_{real} = -0.0009\ pitch_{est}^{2} + 0.9744\ pitch_{est} + 0.6535$	0.99931
Polinómico de grado 3	$pitch_{real} = 2e\text{-}05\ pitch_{est}^{3} - 0.0009\ pitch_{est}^{2} + 0.9615\ pitch_{est} + 0.6461$	0.99933
Polinómico de grado 4	$pitch_{real} = 1.49e\text{-}05\ pitch_{est}^{4} + 8.68e\text{-}07\ pitch_{est}^{3} - 0.016\ pitch_{est}^{2} + 0.969\ pitch_{est} + 1.602$	1

(a)

Tipo de modelo	Ecuación	R^2
Lineal	$roll_{real} = 0.969\ roll_{est} + 0.072$	0.99998
Polinómico de grado 2	$roll_{real} = -0.00012\ roll_{est}^{2} + 0.9694\ roll_{est} + 0.1243$	0.99999
Polinómico de grado 3	$roll_{real} = 1e\text{-}06\ roll_{est}^{3} - 0.0001\ roll_{est}^{2} + 0.9685\ roll_{est} + 0.1242$	0.99999
Polinómico de grado 4	$roll_{real} = 2e\text{-}06\ roll_{est}^{4} + 1e\text{-}06\ roll_{est}^{3} - 0.0023\ roll_{est}^{2} + 0.9685\ roll_{est} + 0.2561$	1

(b)

Tabla 6. *Resumen de la formulación matemática de los modelos estudiados así como sus coeficientes de determinación para el pitch (a) y para el roll (b).*



(a) *(b)*

Figura 24. *Representación gráfica de cómo ajustan los distintos modelos de calibración para el pitch (a) y para el roll (b). Los datos experimentales se representan mediante un asterisco de color rojo, la estimación lineal media trazo discontinuo de color morado, y las estimaciones polinómicas de grado 2, 3 y 4 de color azul, verde y amarillo respectivamente.*

4.2.4 Yaw

4.2.4.1 Cálculo de la observación

En este caso el sensor ya nos es tan preciso como los del pitch y el yaw ya que está sujeto a una deriva significante a lo largo del tiempo como se ha comentado en el apartado 3.3.2. Es por ello que no se puede tratar como una observación directa y se maneja del mismo modo que las medidas de altura relativa. La información sobre el ángulo de giro en z es calculada por el giroscopio correspondiente de la IMU del AR Drone y es enviada con una frecuencia de 200 Hz (5 ms) a través de una wireless LAN siendo interpretada por el driver implementado en ROS que ya se ha descrito anteriormente en el apartado 3.1. Todo este proceso aparece reflejado en la figura 10.

El cálculo del yaw que se introduce en el filtro como observación en un determinado instante de tiempo se realiza como sigue:

$$\Psi_{filtro} = \Psi(t-\delta t) + \hat{\Psi}(t) - \hat{\Psi}(t-\delta t) \qquad [21]$$

Dónde:

- $\Psi(t-\delta t)$ es el ángulo yaw del dron de acuerdo con el filtro tras la última observación del mismo .

- $\hat{\Psi}(t) = \hat{\Psi}_{nav} - baseline_{\Psi}$ es la yaw relativo medida en la iteración actual. Esta magnitud se computa como la diferencia entre la lectura del yaw realizada por el AR Drone (que denotamos como $\hat{\Psi}_{nav}$ que da un valor absoluto del yaw del dron) y una línea de base (que denotamos como $baseline_{\Psi}$, y que básicamente es el valor del yaw que consideramos como 0 del filtro. Este es especialmente importante, ya que el giroscopio puede dar un valor distinto de 0 en la orientación inicial del cuadricóptero (por ejemplo para este dispositivo nuestro 0 podría equivaler a 157º). Cuando $\hat{\Psi}(t) - \hat{\Psi}(t-\delta t) > 2^{o}$ se establece una actualización del al valor de en ese instante.

- $\hat{\Psi}(t-\delta t)$ es el yaw relativo medido en la iteración anterior, es decir, hace 5 ms.

4.2.4.2 Optimización de las observaciones

Debido a los problemas que presenta este sensor relacionados con la deriva, se decidió llevar a cabo un estudio para cuantificar la misma y tratar de reducirla lo máximo posible. Se realizó un proceso de toma de datos sobre las lecturas del yaw durante una maniobra simple de despegue del cuadricóptero, es decir, que partiendo del reposo en el

suelo se le ordena despegar y mantenerse en la posición $(x,y,z,\Psi)^T = (0,0,1,0)^T$. Lo resultados de la maniobra se muestran en la figura 25. En ella se observa claramente un salto en las lecturas proporcionadas por el giroscopio en z de la IMU. Dicho salto coincide con la maniobra de despegue del cuadricóptero.

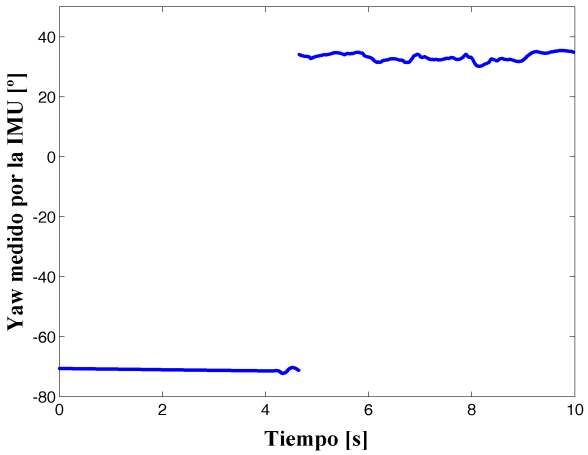


Figura 25. *Representación de las lecturas del ángulo de giro en z (yaw) enviadas por el giroscopio correspondiente de la IMU frente al tiempo para una maniobra en la que el cuadricóptero pasa de estar parado a ser despegado (en torno a los 4 segundos) y mantener una posición de (x,y,z,Ψ) = (0,0,1,0).*

La observación de este fenómeno motivó el estudio en mayor profundidad de cada una de las dos ramas que separa este salto, es decir, el comportamiento del giroscopio antes del despegue y después del mismo. En primer lugar se recopilaron datos sobre las lecturas procesadas con el cuadricóptero sobre el suelo sin ser despegado durante un minuto aproximadamente, lo que se representa en la figura 26.a. A continuación repitió el mismo proceso de toma de medidas pero con el cuadricóptero despegado y manteniéndose en la posición $(x,y,z,\Psi)^T = (0,0,1,0)^T$, lo que se representa en la figura 26.b.

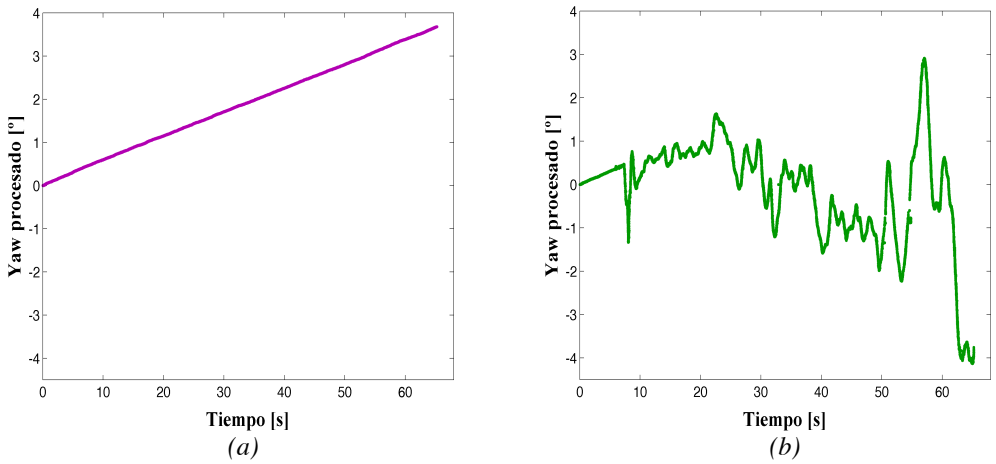


Figura 26. *Representación de las observaciones del yaw introducidas al filtro frente al tiempo. (a) Situación correspondiente al dron en el suelo sin despegar. (b) Situación correspondiente al dron manteniendo la posición (x,y,z,Ψ) =(0,0,1,0) tras el despegue.*

Si observamos la gráfica 26.a se observa claramente que las lecturas del yaw que se introducen al filtro cuando el cuadricóptero está quieto en el suelo están sujetas a una deriva elevada, de en torno a 0.06 º/s, lo que supone que, si por algún motivo, se tarda un minuto en despegar el cuadricóptero, el estado del mismo en lugar de mantenerse a 0º estará en unos 4º. Para corregir este problema, se ha planteado un proceso de filtrado basado en un contador interno. Empleando una utilidad de ROS que nos permite medir el tiempo del sistema, se guarda como referencia de tiempos el primer paquete de altura enviado por el AR Drone que contiene una altura distinta de cero (lo que implica el inicio del despegue) y a partir de ahí se esperan 4.5 segundos antes de poner a funcionar el cálculo del yaw. Este valor se ha determinado empíricamente tras la realización de un gran número de experimentos, comprobándose que los valores enviados por el firmware del AR Drone tras la espera son lecturas correctas del yaw. Para comprobar el correcto funcionamiento del filtro se ha planteado el siguiente experimento: se ha hecho despegar al AR Drone, mantenerse en la posición $(x,y,z,\Psi)^T = (0,0,1,0)^T$ durante un cierto tiempo y posteriormente girar 90º hasta la posición $(x,y,z,\Psi)^T = (0,0,1,90)^T$ (ambos ángulos señalados mediante línea negra de trazo discontinuo).

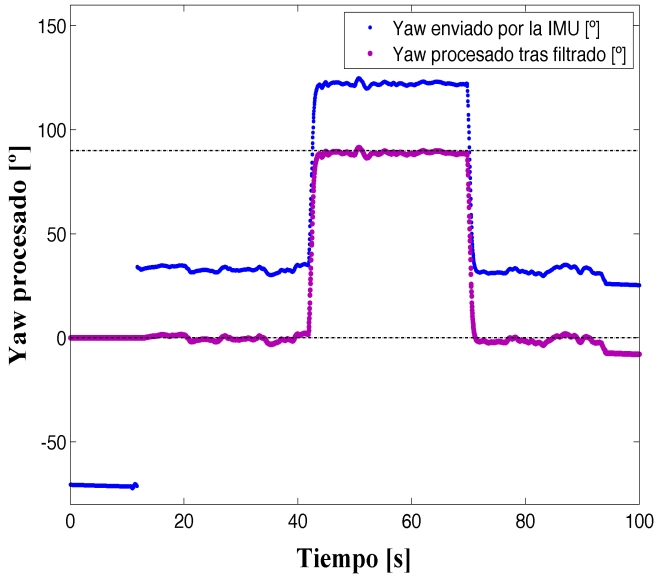


Figura 27. *Representación de las lecturas enviadas por el giroscopio en z de la IMU (color azul) y de las observaciones procesadas que se introducen en el filtro (color morado) con respecto al tiempo en una maniobra que consiste en despegar, mantenerse en la posición (x,y,z,Ψ) = (0,0,1,0) durante un cierto tiempo y posteriormente girar 90º hasta la posición (x,y,z,Ψ) = (0,0,1,90)*

En la figura 27 se han representado los resultados de los experimentos: en color morado se ha representado el yaw que se introduce al filtro tras el procesado frente al tiempo, mientras que en azul se han representado las lecturas del giroscopio en z de la IMU. Se observa que hasta en torno a los 20 segundos (instante en el que ya ha despegado y se ha ejecutado el filtro de tiempos) la lectura introducida el filtro permanece en 0, que es el valor real que nos interesa introducir. Posteriormente las observaciones coinciden

exactamente con las lecturas del giroscopio (pero escaladas según lo que se ha comentado en el apartado 4.2.4.1), incluso para las maniobras de giro.

Por otro lado, no se ha encontrado ningún criterio que funcionase para filtrar la deriva en el yaw durante el vuelo (figura 26.b), ya que esta carece de la tendencia lineal que presenta cuando el cuadricóptero está parado, y más bien se trata de oscilaciones en torno al valor verdadero cuya amplitud se va haciendo más grande con el tiempo de vuelo. La única forma de mantener la deriva en a valores admisibles es empleando el sistema de marcadores artificiales April Tags que se presentan en el apartado 4.3.

4.3 Creación de un sistema de localización en base al sistema de marcadores artificiales April Tags.

Como se ha comentado anteriormente, a pesar de las mejoras realizadas en el proceso de optimización de los datos enviados por los sensores onboard, la navegación del AR Drone durante largos periodos de tiempo sigue sin ser satisfactoria debido a la deriva inherente a los procesos de odometría que se llevan a cabo, y a sensores imprecisos como el giroscopio que mide el giro en z, y que presenta una deriva importante en la medición de dicho ángulo. Se hace necesario, por tanto, incluir un sistema que mejore la navegación y que nos proporcione unas referencias absolutas con las que estimar la posición del dron. Como se ha comentado anteriormente, se ha decidido emplear un sistema basado en marcadores artificiales conocido como April Tags Fiducial System.

En este apartado se explican los procedimientos que se han implementado para transformar las lecturas de dicho sistema en un "sensor" fiable del estado del cuadricóptero. Dicho procesado se ha representado en la figura 2 y se puede dividir en tres operaciones fundamentales: la primera de ellas se corresponde con la realización de las transformaciones algebraicas que nos permitan convertir las lecturas del April Tags Fiducial System proporcionadas en el sistema cámara a una lectura que represente la posición y orientación del cuadricóptero en el sistema de referencia mundo. Esto se presenta en el apartado 4.3.1. En segundo lugar, se debe realizar un proceso de filtrado y calibrado de las lecturas, lo que se presenta en el apartado 4.3.2.

4.3.1 Transformación de las lecturas del April Tags Fiducial System en una observación de la posición y orientación del AR Drone.

En este primer apartado va a detallarse el proceso que se ha desarrollado para transformar las lecturas proporcionadas por el April Tags Fiducial System en una observación de la posición y orientación del AR Drone. En el apartado 4.3.1.1 se expone el desarrollo analítico general de cómo se ha realizado esta transformación, mientras que en el apartado 4.3.1.2 se aborda un aspecto concreto de este proceso de transformación, relacionado con el ajuste de la matriz de rotación de la cámara.

4.3.1.1 Planteamiento general.

En primer lugar se han definido los sistemas de coordenadas que utilizan las distintas fuentes de información (April Tags Fiducial System y sensores onboard) junto con un sistema de referencia absoluto ligado al escenario y las transformaciones que los relacionan. Esto permitirá realizar la fusión de las lecturas procedentes de diferentes fuentes y expresar el estado en un sistema de referencia fijo que es en el que se expresan los estados objetivo (*targets*) así como los diferentes comandos de navegación. Existen por tanto tres sistemas de referencia (ver figura 28), a saber:

- **S_0:** es el sistema de coordenadas fijo ligado al punto de partida. Está compuesto por el origen **O_0**, definido por el usuario y que frecuentemente coincide con el punto de despegue, y por la base **B_0**. La orientación de dicha base está determinada por el criterio original que establecieron los diseñadores del TUM AR Drone, que ya se había comentado anteriormente en la figura 6. Sin embargo, se consideran los ángulos de Euler positivos en el sentido negativo de los ejes principales.

- **S_T:** es el sistema de coordenadas fijo ligado al punto de origen de tag. Está compuesto por dicho punto **O_T**, que coincide con el centro geométrico del tag y que interesa expresar en el sistema S_0, y por la base **B_T.** Las coordenadas de O_{Ta} dependerán, obviamente, del lugar donde coloquemos el tag y la orientación de la base se ha establecido arbitrariamente, tratando de definirla de modo que se simplifiquen los cálculos lo máximo posible. En términos generales, esto consiste en definir el eje *z* perpendicular al plano del tag y hacia dentro, el eje *y* hacia abajo y el *x* hacia la derecha mirando el tag de frente. Los ángulos de Euler se definen en sentido negativo de los ejes principales.

- **S_C:** es el sistema de referencia móvil ligado a la cámara del AR Drone y en el que se expresa la salida del sistema April Tags conteniendo la posición y orientación del tag con respecto a la cámara. Está compuesto por el origen **O_C**, que coincide con la posición de la cámara del AR Drone con respecto a **O_0,** y por la base **B_C**, cuya orientación exacta se ha determinado experimentalmente estudiando la matriz de rotación que devuelve el April Tags Fiducial System en cada instante**.** Los ángulos de Euler se definen en sentido negativo de los ejes principales.

- **S_M:** es el sistema de referencia móvil ligado al centro geométrico del AR Drone. Está compuesto por el origen **O_M**, que coincide con la posición del centro del AR Drone con respecto a **O_0,** y por la base **B_M**, cuya orientación se ha establecido de tal forma que coincida inicialmente con B_0. Sin embargo, una vez el AR Drone haya despegado ya no será así, ya que el sistema experimentará

translaciones y rotaciones. Los ángulos de Euler se definen en sentido negativo de los ejes principales.

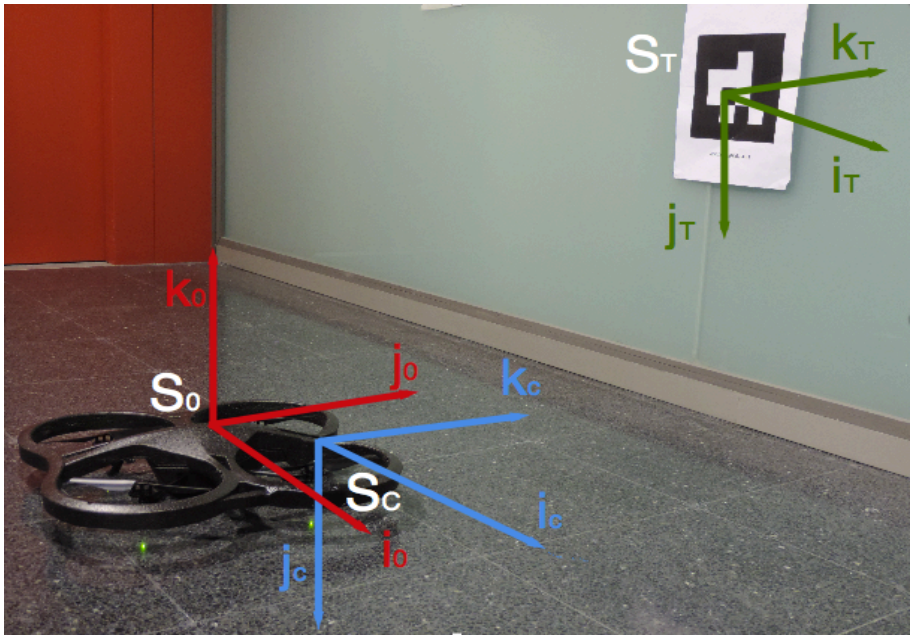


Figura 28. *Representación de los distintos sistemas de coordenadas.*

Una vez establecidos los distintos sistemas de coordenadas se buscará realizar el cálculo de la posición y orientación del dron en el sistema absoluto a partir de los datos suministrados por el April Tags Fiducial System en el sistema cámara, es decir, relacionar el sistema S_C y el S_0 con las matrices de rotación y traslación correspondientes. En primer lugar debemos de ser capaces de expresar la base B_T en la B_0. La primera esta definida de tal forma que se puede escribir:

$$B_T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix} = \begin{pmatrix} i_T \\ j_T \\ k_T \end{pmatrix} \qquad [22]$$

$$B_T = M_T^0 \; B_0 \qquad [23]$$

Siendo M_T^0 la matriz de rotación del sistema B_T con respecto a B_0. Por lo tanto las coordenadas del origen del tag en la base B_0 será:

$$O_T = (x_T^0, y_T^0, z_T^0)\, B_0 = C_T^0 \; B_0 \qquad [24]$$

A continuación se estudiará como se expresa la base del tag en el sistema cámara. En primer lugar hay que considerar la situación inicial en la que el cuadricóptero aún no ha despegado. En este caso la expresión de B_T con respecto a B_C será:

$$B_T^{inicial} = M_T^C \; B_C \qquad [25]$$

Donde la matriz $M_T^C = I$ debido a que los ejes del tag se han definido estratégicamente coincidentes con los de la cámara así para simplificar los cálculos (aunque en el apartado 4.2.1.2 se detalla que esto no es exactamente así). En segundo lugar hay que contemplar el caso del despegue del cuadricóptero y por tanto, de que la cámara haya comenzado a desplazarse. En este caso se obtendrá que:

$$B_T = M_{AT} \; B_T^{inicial} = M_{AT} \; M_T^C \; B_C = M_{AT} \; B_C \qquad [26]$$

Donde la matriz M_{AT} es la que devuelve el April Tags Fiducial System y representa el cambio de orientación de los ejes del tag con respecto a B_C.

También nos va a interesar tener la transformación de la base B_C a la base B_M, que resultará como sigue:

$$B_C = M_C^M \; B_M \qquad [27]$$

Donde la matriz M_C^M es la matriz de rotación del sistema B_C con respecto a B_M y según como se han definido las orientaciones de dichas bases resulta como sigue:

$$M_C^M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \qquad [28]$$

Nótese que $M_C^M = M_T^0$ porque la orientación de las bases B_M y B_0 así como las bases B_C y B_M coinciden.

El siguiente paso será expresar la base del sistema móvil B_M en el sistema de referencia fijo. La expresión resultante tiene la siguiente forma:

$$B_M = M_M^0 \; B_0 \qquad [29]$$

Donde M_M^0 es la matriz de rotación de la base B_M con respecto a B_0. El origen del sistema móvil expresado en el fijo será:

$$O_M = C_M^0 \; B_0 \qquad [30]$$

Llegados a este punto, todas las matrices son conocidas menos la matriz M_M^0. La estimación de la orientación del cuadricóptero consiste fundamentalmente en estimar el

valor de dicha matriz a partir de la matriz M_{AT}, que cambia en función de la posición relativa de la cámara y el tag. La matriz M_M^0 es de la forma:

$$M_M^0 = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \quad [31]$$

Se trata de una matriz de rotación general, y por tanto, representa rotaciones en cada uno de los tres ejes:

$$M_M^0 = M_{M_z}^0(\gamma)\ M_{M_y}^0(\beta)\ M_{M_x}^0(\alpha) \quad [32]$$

$$M_M^0 = \begin{bmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{bmatrix}$$

[33]

Los ángulos de giro pueden obtenerse de la matriz de rotación según las siguientes relaciones:

$$\alpha = \text{atan2}(m_{32}, m_{33}) \quad [34]$$

$$\beta = \text{atan2}(-m_{31}, \sqrt{m_{32}^2 + m_{33}^2}) \quad [35]$$

$$\gamma = \text{atan2}(m_{21}, m_{11}) \quad [36]$$

El valor de esta matriz puede expresarse en base a lo expuesto anteriormente ya que se cumple lo siguiente:

$$B_T = M_{AT}\ B_C = M_{AT}\ M_C^M\ B_M = M_{AT}\ M_C^M\ M_M^0\ B_0 \quad [37]$$

$$B_T = M_T^0\ B_0 \quad [38]$$

Conocemos M_{AT} que es la matriz de rotación que nos devuelve el April Tags Fiducial System, la matriz M_C^M (ver ecuación 28) y la matriz M_T^0 (ver ecuación 23), por lo que si igualamos las expresiones 37 y 38 resulta lo siguiente:

$$M_{AT}\ M_C^M\ M_M^0\ B_0 = M_T^0\ B_0 \quad [39]$$

$$M_{AT}\, M_C^M\; M_M^0 = M_T^0 \qquad [40]$$

$$\boxed{M_M^0 = \left(M_C^M\right)^{-1}\left(M_{AT}\right)^{-1} M_T^0} \qquad [41]$$

Es importante destacar que, al contrario de lo que se podría suponer, se ha observado que el criterio de asignación del sentido de los giros de los ángulos de navegación es bastante arbitrario (ver figura 7). Es por esto que resulta importante prestar atención a como están definidos en las formulaciones de cualquiera de las fuentes de información (sensores onboard y April Tag Fiducial System), siendo necesario realizar ciertas transformaciones adicionales tales como las siguientes:

$$\Phi = -\beta \qquad [42]$$

$$\Theta = -\alpha \qquad [43]$$

$$\Psi = \gamma \qquad [44]$$

En lo que respecta a las traslaciones sabemos que:

$$O_T = O_M + O_C^M + O_T^C \qquad [45]$$

Donde:

- O_T son las coordenadas del centro del tag expresadas en el sistema S_0.
- O_M son las coordenadas del centro geométrico del dron en el sistema S_0.

- O_C^M son las coordenadas del centro de la cámara en el sistema S_M, es decir, el vector $\overline{O_M O_C}$. Este vector puede expresarse como:

  $$O_C^M = C_C^M\; B_M = C_C^M\; M_M^0\; B_0 \qquad [46]$$

  Siendo C_C^M las coordenadas de la cámara del dron con respecto al sistema S_M. Debido a que la cámara está unida rígidamente al resto de dron, está distancia tiene un valor fijo igual a: $(x,y,z)_M = (0,\ 0.2,\ 0)$.

- O_T^C son las coordenadas del centro del tag con respecto al centro de la cámara, es decir, el vector $\overline{O_C O_T}$. Este vector puede expresarse como:

  $$O_T^C = C_T^C\; B_C = C_T^C\; M_C^M\; M_M^0\; B_0 \qquad [47]$$

  Siendo C_T^C las coordenadas de la cámara del tag con respecto al sistema S_C que devuelve la librería April Tags Fiducial System.

Lo que interesa es calcular la posición de la cámara del dron en el sistema absoluto C_M^0, a la que puede llegarse si se recupera la ecuación 45 y se incorporan las definiciones que se acaban de dar y se trata de expresar todo en la base B_0:

$$C_T^0\ B_0 = C_M^0\ B_0 + C_C^M\ M_M^0\ B_0 + C_T^C\ M_C^M\ M_M^0\ B_0 \qquad [48]$$

$$\boxed{C_M^0 = C_T^0\ B_0 - C_C^M\ M_M^0\ B_0 - C_T^C\ M_C^M\ M_M^0\ B_0} \qquad [49]$$

Las componentes de este vector se corresponden directamente con las observaciones de las variables de estado de posición *(x, y, z)* debido a que la orientación de los ejes que hemos escogido coincide con el criterio que viene preestablecido en el TUM AR Drone (ver figura 7).

4.3.1.2 Corrección de la posición de la cámara. Ajuste de M_C^M

Tras la realización de experimentos preliminares, se detectó una falta de precisión en la alineación de los ejes del sistema cámara (S_C) con respecto a los ejes del sistema situado en el centro geométrico del cuadricóptero (S_M), lo cual era de esperar teniendo en cuenta que esta alineación se basa en la colocación de la cámara en un alojamiento plástico que está sujeto a pequeñas deformaciones de montajes además de que la tolerancia de fabricación de estas piezas es baja.

Esto se ha observado cuando al colocar el cuadricóptero sobre el suelo del laboratorio los valores para el roll, pitch y yaw devueltos por el April Tags y por los giroscopios de la IMU no coinciden (ver tabla 7):

	Roll	**Pitch**	**Yaw**
April Tags	-2.4736	0,2095	-0,2717
IMU	1,9806	0,7542	0,0000

Tabla 7. *Resumen comparativo de los ángulos roll, pitch y yaw devueltos por el April Tags y por los giroscopios de la IMU para una prueba realizada sobre un suelo plano.*

Los valores obtenidos deberían corresponderse con *(roll, pitch, yaw) = (0,0,0)* y en el caso de la IMU no coinciden exactamente, probablemente, debido a que físicamente la placa donde se encuentran los giroscopios tiene una pequeña inclinación. Sin embargo, es la referencia que determina la orientación del sistema móvil, siendo la desviación de los ángulos devueltos por el April Tags con respecto a estos valores debida a una cierta desviación de la cámara con respecto a esta orientación. Esto quiere decir que los ejes del S_C están ligeramente girados con respecto a los del S_M. Es por eso que la matriz M_C^M

no se corresponde con la que se escribió en la expresión 28, sino que habrá que calcularla en base a datos experimentales.

Para ello se realizaron cinco pruebas en las que se midieron los ángulos devueltos por el April Tags Fiducial System (es decir los ángulos correspondientes a la orientación del tag con respecto a S_C) y los que calcula la IMU al colocar el cuadricóptero en diversas orientaciones, consistentes en cambiar todos los ángulos una pequeña cantidad (entre 0 y 10º), para que los errores no se debiesen realmente a una mala estimación derivada de no ver bien el tag y para ninguno de los elementos de las matrices de transformación correspondientes a estos ángulos fuese exactamente 0. Una vez obtenidos los datos experimentales se aplica un ajuste por mínimos cuadrados sobre los datos para obtener la mejor estimación posible de la matriz M_C^M .

Para plantear el ajuste hay que partir de la ecuación 40, que puede reescribirse como siguiente:

$$M_{AT}\ M_C^M = M_T^0\ \left(M_M^0\right)^{-1} \qquad [50]$$

En base a esta igualdad se pueden plantear una serie de sistemas determinados tales como:

$$\left.\begin{array}{l} A^1\ x_1^1 = B_1^1 \\ A^1\ x_2^1 = B_2^1 \\ A^1\ x_3^1 = B_3^1 \end{array}\right\}\ \left.\begin{array}{l} A^2\ x_1^2 = B_1^2 \\ A^2\ x_2^2 = B_2^2 \\ A^3\ x_3^2 = B_3^2 \end{array}\right\}\ ....\ \left.\begin{array}{l} A^j\ x_1^j = B_1^j \\ A^j\ x_2^j = B_2^j \\ A^j\ x_3^j = B_3^j \end{array}\right\} \qquad [51]$$

Donde:

- A^j es igual a M_{AT}, es decir, la matriz de transformación correspondiente a los ángulos de Euler devueltos por el April Tags Fiducial System, para la prueba *j* (en el caso de estudio *j=1,2,3,4,5*).

- B_1^j, B_2^j, B_3^j son las columnas 1, 2 y 3 de la matriz B, es decir, de la matriz resultante de multiplicar M_T^0 (cuya expresión viene dada por la ecuación 23) por la inversa de la matriz de transformación correspondiente a los ángulos de Euler devueltos por la IMU $\left(M_M^0\right)^{-1}$, que equivale a la matriz de transformación del sistema móvil (S_M) al sistema S_O, para la prueba *j* (en el caso de estudio *j=1,2,3,4,5*).

- x_1^j, x_2^j, x_3^j son las columnas 1, 2 y 3 de la matriz M_C^M, es decir, de la matriz de transformación del sistema cámara (S_C) al sistema móvil (S_M) para la prueba *j* (en el caso de estudio *j=1,2,3,4,5*).

En lo que sigue, se va plantear una combinación de los resultados de las distintas pruebas en un único sistema sobredeterminado, que se tratará de resolver por el método de mínimos cuadrados.

Dicho sistema combinará las matrices A^j obtenidas para cada prueba en una supermatriz de dimensiones 15x3, y las columnas B_1^j, B_2^j, B_3^j de cada prueba para dar lugar a un supervector de dimensiones 15x1. Se calculan de este modo las columnas de la matriz M_C^M de un modo en el que se combinan todas las pruebas realizadas. El sistema en cuestión se muestra en la ecuación 52:

$$\left.\begin{aligned} A\ M_1 &= \mathrm{B}_1 \\ A\ M_2 &= \mathrm{B}_2 \\ A\ M_3 &= \mathrm{B}_3 \end{aligned}\right\} \qquad [52]$$

Donde:

$$A = \begin{bmatrix} A^1 \\ A^2 \\ A^3 \\ A^4 \\ A^5 \end{bmatrix}_{15x3} \quad B_1 = \begin{bmatrix} B_1^1 \\ B_1^2 \\ B_1^3 \\ B_1^4 \\ B_1^5 \end{bmatrix}_{15x1}, \ B_2 = \begin{bmatrix} B_2^1 \\ B_2^2 \\ B_2^3 \\ B_2^4 \\ B_2^5 \end{bmatrix}_{15x1}, \ B_3 = \begin{bmatrix} B_3^1 \\ B_3^2 \\ B_3^3 \\ B_3^4 \\ B_3^5 \end{bmatrix}_{15x1}$$

Para resolver por mínimos cuadrados hay que reescribir el sistema del siguiente modo:

$$\left.\begin{aligned} \left(A^T A\right) M_1 &= A^T\ \mathrm{B}_1 \\ \left(A^T A\right) M_2 &= A^T\ \mathrm{B}_2 \\ \left(A^T A\right) M_3 &= A^T\ \mathrm{B}_3 \end{aligned}\right\} \qquad [53]$$

Llegados a este punto resta imponer una última condición de ortonormalidad. Para obtenerla se ha diseñado el siguiente procedimiento:

1. En primer lugar hay que definir los vectores que conforman la base a partir de la matriz obtenida:

$$M_C^M = \begin{bmatrix} 1 & 0.0054 & 0.0055 \\ -0.0088 & 0.0534 & -0.9985 \\ -0.0058 & 0.9986 & 0.0524 \end{bmatrix} \begin{matrix} \rightarrow v_1 \\ \rightarrow v_2 \\ \rightarrow v_3 \end{matrix} \quad [54]$$

2. En segundo lugar se deben normalizar los vectores dividiéndolos por su módulo:

$$\hat{v}_1 = \frac{v_1}{|v_1|} \qquad \hat{v}_2 = \frac{v_2}{|v_2|} \qquad \hat{v}_3 = \frac{v_3}{|v_3|} \quad [55]$$

3. A continuación se construye una base ortonormal a partir de los vectores $\hat{v}_1$, $\hat{v}_2$ y $\hat{v}_3$. Para ello se necesita llevar a cabo lo siguiente:

   3.1 El primer vector de la base lo definimos como: $\hat{S} = \hat{v}_1 + \hat{v}_2$ [56]

   3.2 El segundo vector de la base lo obtenemos como: $\hat{n} = \hat{v}_1 \times \hat{S}$ [57]

   3.3 El tercer vector de la base se calcula como: $\hat{p} = \hat{n} \times \hat{S}$ [58]

4. Por último se proyectan los vectores $\hat{v}_1$, $\hat{v}_2$ y $\hat{v}_3$ en esta nueva base ortonormal:

$$\hat{v}_{1*} = \hat{S}\ \cos 45^\circ\ +\ \hat{p}\ \sin 45^\circ \quad [59]$$

$$\hat{v}_{2*} = \hat{S}\ \cos 45^\circ\ -\ \hat{p}\ \sin 45^\circ \quad [60]$$

$$\hat{v}_{3*} = \hat{v}_{1*} \times \hat{v}_{2*} \quad [61]$$

El proceso se ha explicado con los vectores (v_1, v_2) pero también se ha aplicado a las combinaciones (v_2, v_3) y (v_3, v_1). Para estas distintas combinaciones se han obtenidos diferentes matrices de transformación, por lo para calcular cuan buena ha sido la aproximación se establecido dos criterios para el cálculo del error:

1. Computo de la norma del error en cada vector y error total:

$$e_1 = \frac{v_1 - \hat{v}_{1*}}{|v_1 - \hat{v}_{1*}|} \qquad e_2 = \frac{v_2 - \hat{v}_{2*}}{|v_2 - \hat{v}_{2*}|} \qquad e_3 = \frac{v_3 - \hat{v}_{3*}}{|v_3 - \hat{v}_{3*}|} \quad [62]$$

$$e = norm(e_1, e_2, e_3) \quad [63]$$

2. Búsqueda de la máxima diferencia entre elementos al restar matrices:

$$\max\left(M_C^M - M_{C*}^M\right) \quad [64]$$

a) Combinación 1: (v_1, v_2)

- Aproximación:

$$M_C^M = \begin{bmatrix} 1 & 0.0058 & -0.0015 \\ -0.0017 & 0.0534 & -0.9986 \\ -0.0057 & 0.9986 & 0.0534 \end{bmatrix} \begin{matrix} \rightarrow \hat{v}_{1*} \\ \rightarrow \hat{v}_{2*} \\ \rightarrow \hat{v}_{3*} \end{matrix} \quad [65]$$

- Errores:

$$e_1 = 0.007 \; ; \; e_2 = 0.007 \; ; \; e_3 = 0.001 \; \rightarrow e = 0.0099$$
$$\max\left(M_C^M - M_{C*}^M\right) = 0.007$$

b) Combinación 2: (v_2, v_3)

- Aproximación:

$$M_C^M = \begin{bmatrix} 0.9999 & 0.0062 & -0.0084 \\ -0.0088 & 0.0529 & -0.9986 \\ -0.0058 & 0.9986 & 0.0529 \end{bmatrix} \begin{matrix} \rightarrow \hat{v}_{1*} \\ \rightarrow \hat{v}_{2*} \\ \rightarrow \hat{v}_{3*} \end{matrix} \quad [66]$$

- Errores:

$$e_1 = 0.0140 \; ; \; e_2 = 5.0846 \text{ e-}04 \; ; \; e_3 = 5.0846 \text{ e-}04 \; \rightarrow e = 0.0140$$
$$\max\left(M_C^M - M_{C*}^M\right) = 0.0140$$

c) Combinación 3: (v_3, v_1)

- Aproximación:

$$M_C^M = \begin{bmatrix} 1 & 0.0055 & 0.0055 \\ 0.0052 & 0.0524 & -0.9986 \\ -0.0058 & 0.9986 & 0.0524 \end{bmatrix} \begin{matrix} \rightarrow \hat{v}_{1*} \\ \rightarrow \hat{v}_{2*} \\ \rightarrow \hat{v}_{3*} \end{matrix} \quad [67]$$

- Errores:

$$e_1 = 3.6579 \text{ e-}05 \; ; \; e_2 = 0.0140 \; ; \; e_3 = 3.6579 \text{ e-}05 \; \rightarrow e = 0.0140$$
$$\max\left(M_C^M - M_{C*}^M\right) = 0.0140$$

Consecuentemente la mejor aproximación es la primera ya que es la que da los menores errores ($e = 0.0099$ y $\max(M_C^M - M_{C*}^M) = 0.007$).

Si se utiliza la matriz de la ecuación 65 y se repite la prueba de colocar el cuadricóptero sobre un suelo plano, se obtienen los valores expuestos en la tabla 8. Mediante la comparación de estos resultados con los que aparecen en la tabla 7, se observa que el error absoluto entre las medidas de la IMU y el April Tags ha pasado de 0.493º a 0.096º para el roll, de 0.5447º a 0.2041º para el pitch y de 0.2717º a 0.1723º yaw. Se concluye entonces que la nueva matriz funciona correctamente.

	Roll	**Pitch**	**Yaw**
April Tags	1.3822	0.5791	-0,1723
IMU	1.4782	0,7832	0,0000

Tabla 8. *Resumen comparativo de los ángulos roll, pitch y yaw devueltos por el April Tags y por los giroscopios de la IMU para una prueba realizada sobre un suelo plano tras aplicar la matriz* M_C^M *calibrada.*

4.3.2 Estudio de las lecturas del April Tags Fiducial System

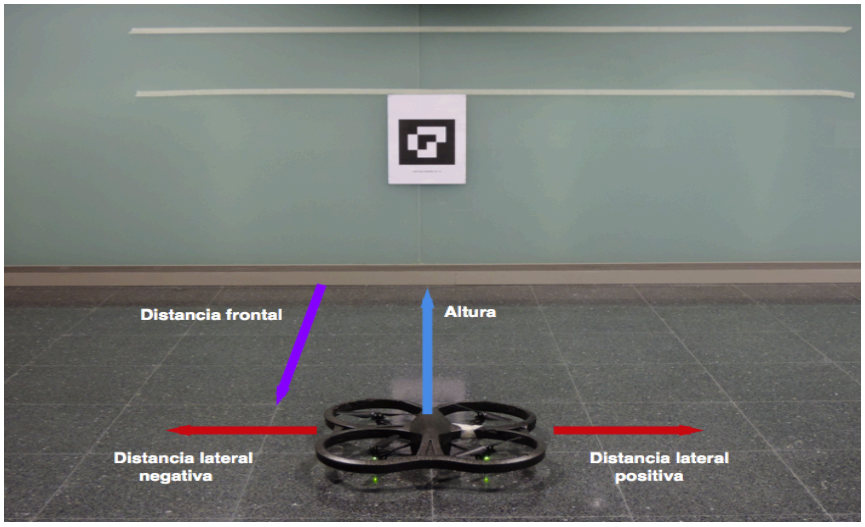
Las librerías del April Tags Fiducial System devuelven, como ya se ha comentado previamente, la posición del tag con respecto al centro de la cámara que los esté observando. Se decidió estudiar estas lecturas a nivel de exactitud y de precisión mediante la realización de procedimiento experimental de toma de datos que se presenta en el apartado 4.3.2.1. El primer concepto se refiere a la dispersión del conjunto de valores obtenidos para las mediciones (siendo esta magnitud inversamente proporcional a la dispersión de los datos), lo que implicó la realización de un proceso de filtrado de las medidas que se presenta en el apartado 4.3.2.2. El segundo concepto hace referencia a cuán cerca está el valor medido del valor real, lo que llevó a realizar un proceso de calibración de las medidas presentado en el apartado 4.3.2.3.

4.3.2.1 Proceso de toma de datos

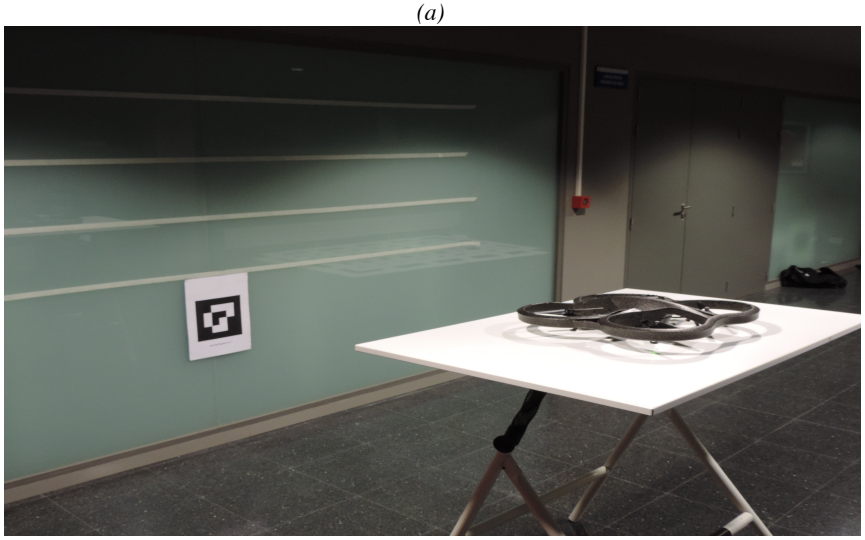
El proceso de obtención de datos para la realización del estudio de precisión y exactitud de las lecturas del April Tag Fiducial System implicó el diseño de un procedimiento experimental que se explica a continuación. Fundamentalmente consiste en colocar un tag en una de las paredes del laboratorio y el AR Drone justo en frente (ver figura 29.a) e ir cambiando las posición y orientación relativa entre ambos. Interesará ir variando las siguientes magnitudes:

- **Distancia frontal:** se refiere a la distancia en metros que existe entre el tag y el centro geométrico del AR Drone en la dirección perpendicular al plano del tag, tal y como se detalla en la figura 29.a.

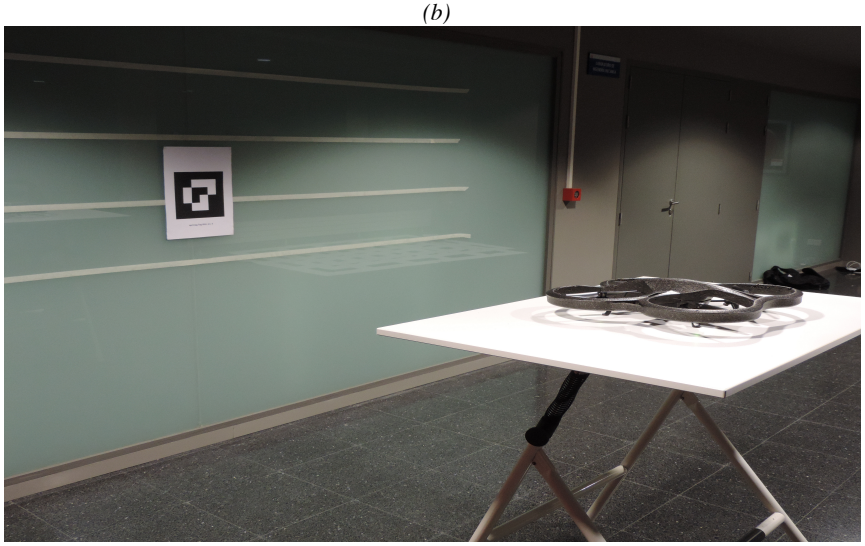
- **Distancia lateral:** se refiere a la distancia en metros que existe entre el tag y el centro geométrico del AR Drone en la dirección paralela al plano del tag tal y



(a)

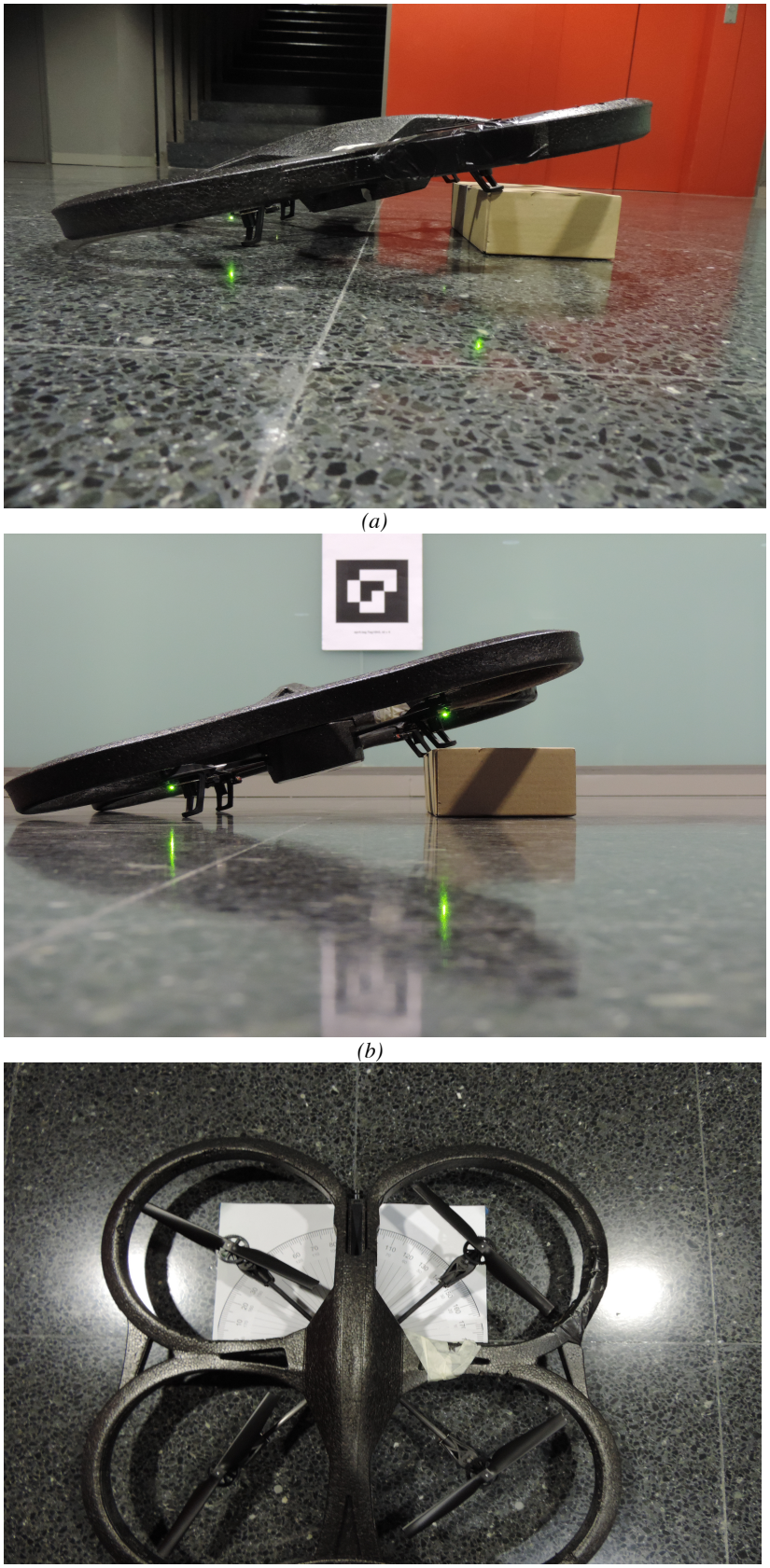


(b)



(c)

Figura 29. *(a) Representación de las magnitudes lineales que se han medido así como la primera posición de altura relativa entre el cuadricóptero (0 m) y el tag (0.53 m). (b) Segunda posición de altura relativa entre el cuadricóptero (1 m) y el tag (0.53 m). (c) Tercera posición de altura relativa entre el cuadricóptero (1 m) y el tag (1.28 m).*



(a)

(b)

(c)

***Figura 30**. Fotografía que ilustra el método utilizado para la medición de los ángulos pitch (a), roll (b) y yaw (c).*

como se detalla en la figura 29.a. La distancia 0 se corresponde con el caso de que el eje de la cámara del AR Drone esté alineado con la dirección normal al centro del tag.

- **Altura AR Drone:** se refiere a la distancia en metros que separamos el AR Drone de la cota del suelo. Cuando el cuadricóptero se encuentra posado sobre el mismo, la distancia se considerará 0 (ver figura 29.a), mientras que si lo posamos sobre una mesa de 1 m de altura, la distancia se corresponderá con dicho valor (figura 29.b y 29.c).

- **Altura ATag:** se refiere a la distancia en metros a la que se ha encuentra el centro del marcador físico April Tag con respecto a la cota del suelo. Las alturas a las que se ha colocado son 0.23 m (figura 29.a), 0.53 m (figura 29.b) y 1.28 m (figura 29.c).

- **Roll**: se corresponde con el ángulo de giro alrededor de la dirección de la distancia frontal. Para medir este ángulo se emplea una caja que genera una inclinación conocida (tras previa medida con un transportador de ángulos) de -16º al apoyar sobre la misma las patas del lado izquierdo del AR Drone (caso representado en la figura 30.b) o de 16º cuando se apoyan las patas del lado derecho sobre la misma.

- **Pitch:** se corresponde con el ángulo de giro alrededor de la dirección de la distancia lateral. Para medir este ángulo se emplea una caja que genera una inclinación conocida (tras previa medida con un transportador de ángulos) de -15º al apoyar sobre la misma las patas delanteras del AR Drone (caso representado en la figura 30.a) o de 15º cuando se apoyan las patas traseras sobre la misma.

- **Yaw:** se corresponde con el ángulo de giro alrededor de la dirección de la altura. Para la medición de este ángulo se ha empleado un transportador. Los giros en sentido horario con respecto a la orientación 0 (caso representado en la figura 30.c) generan ángulos de yaw positivos mientras que los giros en sentido antihorario generan ángulos negativos.

En total se tomaron medidas en 118 posiciones diferentes, cada una de ellas correspondiendo a una fila de la tabla 9 y siendo definidas en función de los parámetros que se acaban de explicar. Es importante recordar que las lecturas que devuelve el April Tags Fiducial System están expresadas en el sistema de la cámara, por lo que cuando se hable de distancias reales no se hace referencia exactamente a las magnitudes que se acaban de explicar, que están definidas en el sistema mundo. Se ha escogido este sistema para explicar el montaje experimental porque facilita la comprensión de cual es

la ubicación real del cuadricóptero. La relación entre este sistema y el sistema cámara viene dada por las transformaciones que se han explicado en el apartado 4.3.1. En cada una de las posiciones se guardó un log de 1500 lecturas del April Tag Fiducial System, que se analizarán en los apartados siguientes a nivel de exactitud y precisión.

Primera batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
1.206	0	0	0.23	0	0	0
1.206	0	0	0.23	-15	0	0
1.206	0	0	0.23	0	-16	0
1.206	0	0	0.23	0	16	0
1.206	0	0	0.23	0	0	20
1.206	-0.402	0	0.23	0	0	0
1.206	-0.402	0	0.23	-15	0	0
1.206	-0.402	0	0.23	0	16	0
1.206	-0.402	0	0.23	0	0	20
1.206	0.402	0	0.23	0	0	0
1.206	0.402	0	0.23	-15	0	0
1.206	0.402	0	0.23	0	0	-16
1.206	0.402	0	0.23	0	0	-20

Segunda batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
1.608	0	0	0.23	0	0	0
1.608	0	0	0.23	-15	0	0
1.608	0	0	0.23	0	-16	0
1.608	0	0	0.23	0	16	0
1.608	0	0	0.23	0	0	-20
1.608	0	0	0.23	0	0	20
1.608	-0.402	1	1.28	0	0	0
1.608	-0.402	1	1.28	-15	0	0
1.608	-0.402	1	1.28	0	16	0
1.608	-0.402	1	1.28	0	0	20
1.608	-0.402	1	1.28	0	0	45
1.608	0.402	1	1.28	0	0	0
1.608	0.402	1	1.28	-15	0	0
1.608	0.402	1	1.28	0	-16	0
1.608	0.402	1	1.28	0	16	0
1.608	0.402	1	1.28	0	0	-20

Tercera batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
2.01	0	0	0.23	0	0	0
2.01	0	0	0.23	-15	0	0
2.01	0	0	0.23	0	-16	0
2.01	0	0	0.23	0	16	0

2.01	0	0	0.23	0	0	20
2.01	0	1	0.53	0	0	0
2.01	0	1	0.53	0	-16	0
2.01	0	1	0.53	0	16	0
2.01	0	1	0.53	0	0	20
2.01	0	1	0.53	0	0	-20
2.01	0	1	1.28	0	0	0
2.01	0	1	1.28	-15	0	0
2.01	0	1	1.28	0	-16	0
2.01	0	1	1.28	0	16	0
2.01	0	1	1.28	0	0	20
2.01	0	1	1.28	0	0	-20

Cuarta batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
2.412	0	0	0.23	0	0	0
2.412	0	0	0.23	-15	0	0
2.412	0	0	0.23	0	-16	0
2.412	0	0	0.23	0	16	0
2.412	0	0	0.23	0	0	20
2.412	0	0	0.23	0	0	-20
2.412	-0.402	0	0.23	0	0	0
2.412	-0.402	0	0.23	-15	0	0
2.412	-0.402	0	0.23	0	-16	0
2.412	-0.402	0	0.23	0	16	0
2.412	-0.402	0	0.23	0	0	20
2.412	-0.402	0	0.23	0	0	45
2.412	-0.804	0	0.23	0	0	0
2.412	-0.804	0	0.23	-15	0	0
2.412	-0.804	0	0.23	0	-16	0
2.412	-0.804	0	0.23	0	16	0
2.412	-0.804	0	0.23	0	0	20
2.412	-0.804	0	0.23	0	0	45
2.412	0.402	0	0.23	0	0	0
2.412	0.402	0	0.23	-15	0	0
2.412	0.402	0	0.23	0	-16	0
2.412	0.402	0	0.23	0	16	0
2.412	0.402	0	0.23	0	0	-20
2.412	0.804	0	0.23	0	0	0
2.412	0.804	0	0.23	-15	0	0
2.412	0.804	0	0.23	0	-16	0
2.412	0.804	0	0.23	0	16	0
2.412	0.804	0	0.23	0	0	-20
2.412	0.804	0	0.23	0	0	-45
2.412	0	1	0.53	0	0	-20
2.412	0	1	0.53	0	0	0
2.412	0	1	0.53	0	-16	0
2.412	0	1	0.53	0	0	20
2.412	-0.402	1	0.53	0	0	0
2.412	-0.402	1	0.53	0	-16	0
2.412	-0.402	1	0.53	0	16	0
2.412	-0.402	1	0.53	0	0	20
2.412	-0.402	1	0.53	0	0	45

2.412	0.402	1	0.53	0	0	0
2.412	0.402	1	0.53	0	-16	0
2.412	0.402	1	0.53	0	16	0
2.412	0.402	1	0.53	0	0	-20
2.412	0.804	1	1.28	0	0	0
2.412	0.804	1	1.28	-15	0	0
2.412	0.804	1	1.28	0	-16	0
2.412	0.804	1	1.28	0	16	0
2.412	0.804	1	1.28	0	0	-20
2.412	0.804	1	1.28	0	0	-45
2.412	-0.804	1	1.28	0	0	0
2.412	-0.804	1	1.28	-15	0	0
2.412	-0.804	1	1.28	0	-16	0
2.412	-0.804	1	1.28	0	16	0
2.412	-0.804	1	1.28	0	0	20
2.412	-0.804	1	1.28	0	0	45

Quinta batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
2.814	0	0	0.23	0	0	0
2.814	0	0	0.23	-15	0	0
2.814	0	0	0.23	0	-16	0
2.814	0	0	0.23	0	16	0
2.814	0	0	0.23	0	0	20
2.814	0	0	0.23	0	0	-20
2.814	0	1	0.53	0	0	0
2.814	0	1	0.53	0	-16	0
2.814	0	1	0.53	0	16	0
2.814	0	1	0.53	0	0	20
2.814	0	1	0.53	0	0	-20

Sexta batería de medidas

Distancia frontal	Distancia lateral	Altura AR Drone	Altura ATag	Pitch	Roll	Yaw
3.216	0	0	0.23	0	0	0
3.216	0	0	0.23	-15	0	0
3.216	0	0	0.23	0	-16	0
3.216	0	0	0.23	0	16	0
3.216	0	0	0.23	0	0	-20
3.216	0	0	0.23	0	0	20
3.216	0	1	0.53	0	0	0
3.216	0	1	0.53	0	-16	0
3.216	0	1	0.53	0	16	0
3.216	0	1	0.53	0	0	20
3.216	0	1	0.53	0	0	-20

Tabla 9. *Compendio de todas las medidas tomadas (un total de 118). Para cada una de las posiciones (filas de la tabla) se guardó a su vez un log de 1500 lecturas devueltas por el April Tags Fiducial System.*

4.3.2.1 Filtrado de los datos

Una vez obtenidos los datos mediante el procedimiento experimental explicado, lo primero que hay que llevar a cabo es un estudio de la exactitud de los mismos, ya que se observa que el April Tags Fiducial System devuelve los datos con una dispersión muy elevada.

Un ejemplo ilustrativo de este problema se muestra en la figura 31, correspondiente a una posición de 0.804 m de distancia lateral, 2.412 m de distancia frontal, 0 m de altura del AR Drone, 0.23 m de altura del tag, 0º de pitch, 16º de roll y 0º de yaw. Se representan en color azul los valores sin procesar devueltos por el sistema April Tags y en color morado el resultado de filtrar los datos para las variables distancia lateral (31.a), distancia frontal (31.b), altura del AR Drone relativa al tag (31.c), pitch (31.d), roll (31.e) y yaw (31.f). Los valores reales en el sistema cámara son respectivamente: -0.790 m , 2.216 m, -0.357 m, 0.242º, 0.535º y 18.06º.

Esta prueba supone un caso extremo en el que se observa claramente que, además del problema de la dispersión, también aparece un sesgo de valores. Esto último se observa perfectamente en la gráfica 31.a, por ejemplo (correspondiente a la distancia lateral). En ella se observa que hay un gran número de observaciones en torno a -0.7 m (muy próximas al valor real de -0.79 m) pero también hay muchas otras en torno a -0.4 m.

Se hace pues imprescindible diseñar un proceso de filtrado de los datos que permita reducir al máximo esta dispersión y eliminar este sesgo de valores que se observa. El filtro creado se explica de forma general en el diagrama de flujo que se muestra en la figura 32, donde:

- $\mathbf{x_j}$ es la lectura del April Tags Fiducial que consideramos, pudiendo corresponderse con la distancia frontal, distancia lateral, altura relativa entre el AR Drone y el tag, el roll, el pitch o el yaw. El índice j representa el número de observación, es decir, 1 se refiere a la primera lectura que recibimos del April Tags, la 2 la segunda y así sucesivamente.

- **tol** se refiere a la tolerancia a partir de la cual aplicamos el criterio. Para x, y, z está tolerancia tiene el valor de **0.1 m,** mientras que para roll, pitch, yaw tiene un valor de **5 º.**

- **|x|** se refiere al valor absoluto de x.

- **|&&** es el operador lógico AND.

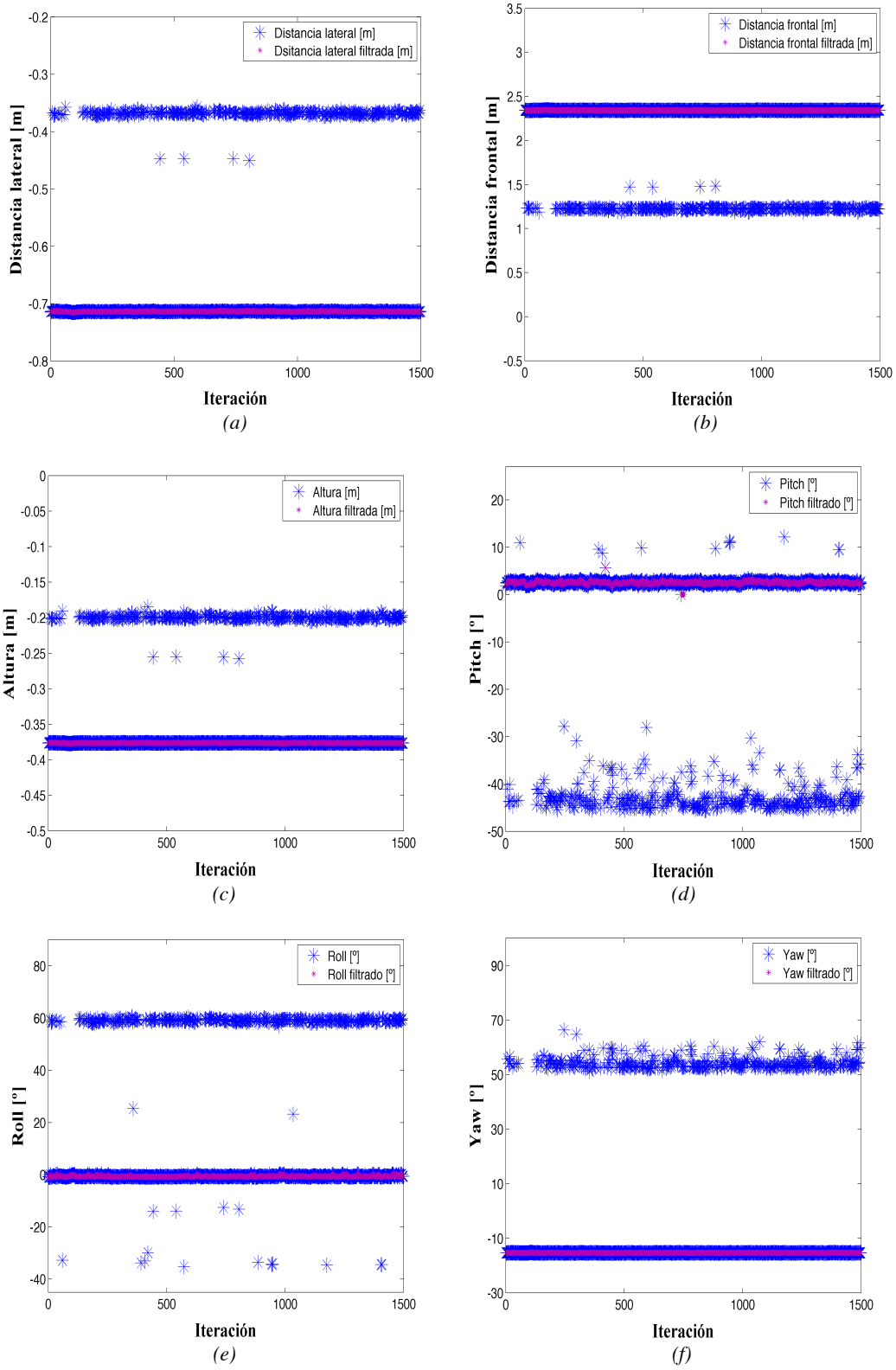


Figura 31. *Representación de las lecturas proporcionadas por el sistema April Tags sin filtrar (en color azul) y tras ser filtradas (color morado) para cada uno de los 1500 valores tomados en la posición real (expresada en el sistema cámara) de: (a) -0.790 m de distancia lateral (b) 2.219 m de distancia frontal, (c) -0.357 de altura del AR Drone relativa al tag, (c) 0.242º de pitch, (d) 0.535º de roll (0.535º) y (f) 18.06º de yaw.*

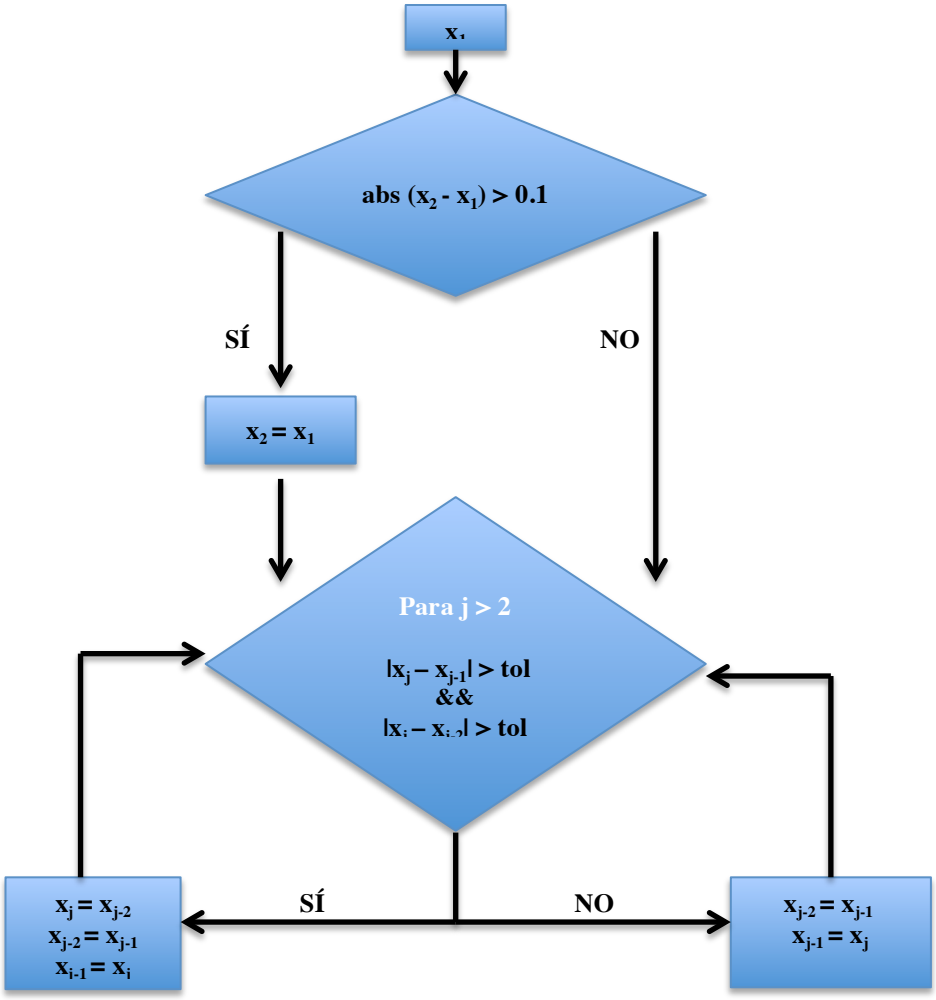


Figura 32. *Diagrama de flujo que explica el procedimiento aplicado para filtrar las lecturas del April Tags Fiducial System.*

La efectividad del filtro descrito se estudia en la figura 33. En ella se compara la desviación típica que existe para cada magnitud (distancia frontal, lateral, altura, roll, pitch y yaw) antes (en color azul) y después (en color rojo) de aplicar el filtrado. La desviación para cada una de las 118 posiciones (ver tabla 9) se ha obtenido a partir del conjunto de 1500 lecturas devueltas por el April Tags en cada una de las mismas.

- Para la distancia lateral (figura 33.a) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 0.02 m, cuando antes de la aplicación del mismo 56 de las 118 posiciones se encontraba por encima de este valor (con un máximo de 0.7 m).

- Para la distancia frontal (figura 33.b) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 0.014 m, cuando antes de la aplicación del mismo 79 de las 118 posiciones se encontraba por encima de este valor (con un máximo de 1.16 m).

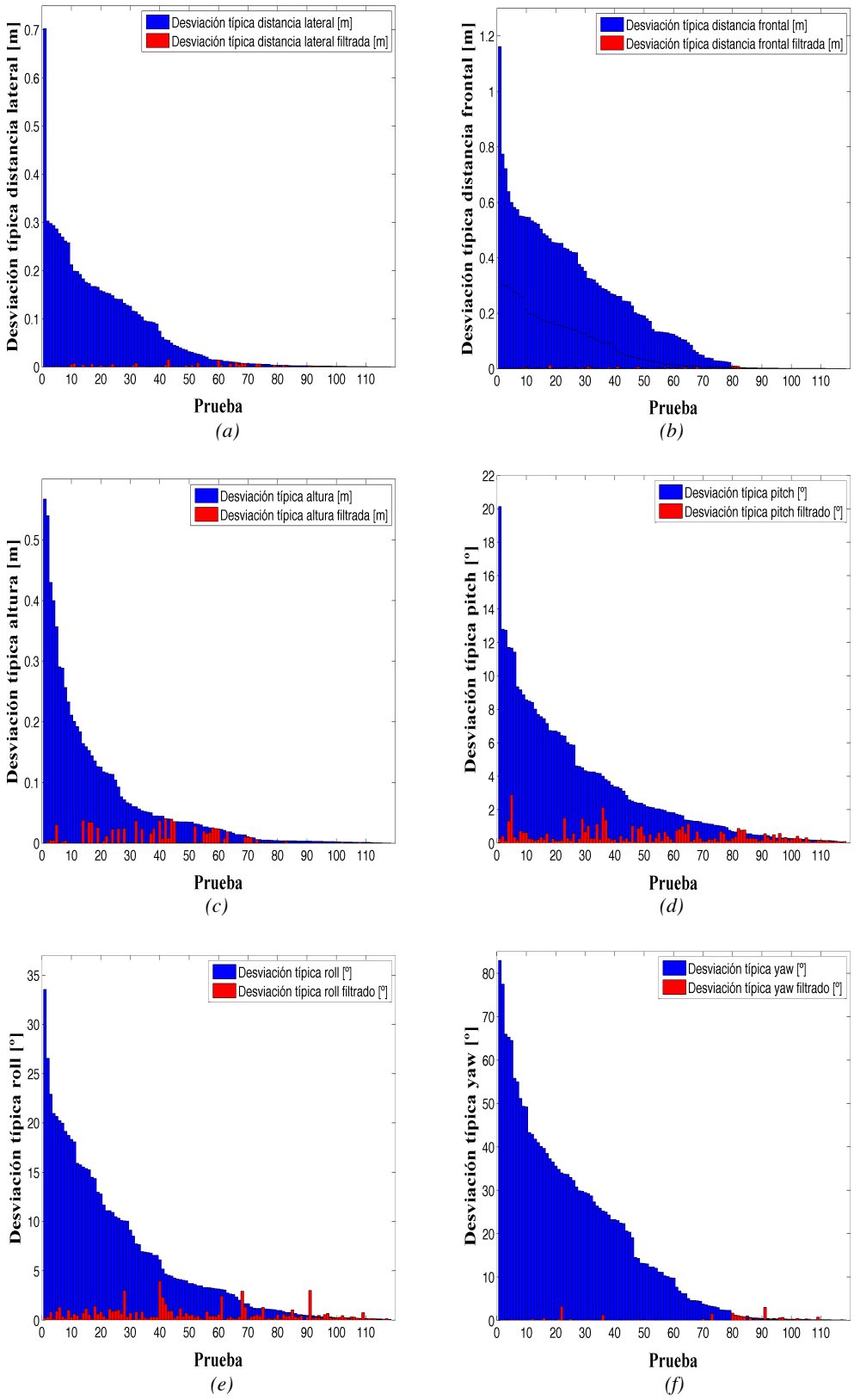


***Figura 33**. Conjunto de gráficas en las que se representa la desviación típica de las lecturas del April Tags Fiducial System antes y después de la aplicación del filtro para cada una de las 118 posiciones representadas en la tabla 9. (a) Distancia lateral. (b) Distancia frontal (c) Altura relativa entre el tag y el AR Drone. (d) Pitch. (e) Roll. (f) Yaw.*

- Para la altura (figura 33.c) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 0.045 cm, cuando antes de la aplicación del mismo 37 de las 118 posiciones se encontraban por encima de este valor (con un máximo de 0.57 m).

- Para el pitch (figura 33.d) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 4º, cuando antes de la aplicación del mismo 36 de las 118 posiciones se encontraban por encima de este valor (con un máximo de 20.13º).

- Para el roll (figura 33.e) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 3º, cuando antes de la aplicación del mismo 62 de las 118 posiciones se encontraban por encima de este valor (con un máximo de 33.53º).

- Para el yaw (figura 33.f) la desviación de los datos se ve reducida en todos los casos tras la aplicación del filtro a un valor inferior a 3.5º, cuando antes de la aplicación del mismo 71 de las 118 posiciones se encontraban por encima de este valor (con un máximo de 83º).

Resulta pues evidente que se ha logrado una notoria mejora en la dispersión para aquellos casos que presentaba desviaciones realmente altas.

4.3.2.3 Calibración de las mediciones

Tras la aplicación del filtrado, se observa persiste un error sistemático que indica la necesidad de un proceso de calibración.

Este consistió en la valoración de diversos modelos de calibración para tratar de ajustar las lecturas lo máximo posible al valor real. Tras probarse varios modelos, se decidió optar por un modelo lineal multivariable debido a la dependencia que se observa entre las distintas variables del problema. Para llevar a cabo la calibración se decidió redefinir la distancia lateral como coeficiente lateral (CL) y la altura como coeficiente de altura (CA):

$$CL = \frac{\text{distancia lateral}}{\text{distancia frontal}} \qquad [68]$$

$$CA = \frac{\text{altura}}{\text{distancia frontal}} \qquad [69]$$

Esto se llevó a cabo porque se consideró que los errores pueden estar asociados a determinadas deformaciones en ciertas partes de la imagen captada por la cámara. La posición de un elemento dentro de la imagen queda definida por estos coeficientes lateral y de altura. Por ejemplo, cuando el cuadricóptero proporcione unas lecturas de distancia lateral = 0.804 m y distancia frontal = 2.412 m (*CL = 0.333*) significa que verá el tag más escorado en una esquina de la imagen, y por tanto, será más probable que de malas estimaciones que cuando el cuadricóptero se encuentra de frente al tag (distancia lateral 0) y a la misma distancia frontal (*CL = 0*).

Lo mismo se puede aplicar a la altura, ya que el cuadricóptero estará viendo peor el tag cuando, por ejemplo, proporciona unas lecturas de altura relativa de -0.23 m y distancia frontal de 2.412 m (*CA = -0.0954*) que cuando se encuentra a la misma lectura de distancia frontal pero una altura de 0.47 m (*CA = 0.1949 m*)

Para realizar la calibración, el objetivo fue minimizar la suma de las diferencias cuadráticas entre el coeficiente lateral, la distancia frontal, el coeficiente de altura, el pitch, el roll y el yaw reales y los estimados con los modelos de calibración (que se detallan en la tabla 10):

$$\min\left(\sum (x_{real} - x_{calibrada})\right) \qquad [70]$$

Modelo $x_{calibrada} = \alpha$ coeficiente lateral $+ \beta$ distancia frontal $+ \gamma$ coeficiente altura $+ \delta$ *pitch* $+ \sigma$ *roll* $+ \lambda$ *yaw* $+ \varepsilon$							
	α	β	γ	δ	σ	λ	ε
Coeficiente lateral	0.96023	0.01513	-0.03492	0.00151	7.326 E-05	0.00041	-0.06054
Distancia lateral	0.00406	0.96159	-0.05478	0.0011	-0.00075	-4.442E-05	-0.04677
Coeficiente altura	-0.00899	0.01025	1.04207	-0.00332	-0.00046	0.00035	0.01456
Pitch	1.42669	-0.20523	5.7298	0.65224	0.04410	-0.01292	4.12011
Roll	-3.54471	-0.56579	1.52771	-0.01291	0.93093	-0.02289	0.61414
Yaw	-0.56471	-0.58384	4.64131	-0.11032	0.01786	0.94175	1.51322

***Tabla 10**. Resumen de los distintos modelos de calibración implementados. El término $x_{calibrada}$ se refiere a la medida estimada por el modelo de calibración, es decir, el coeficiente lateral, la distancia lateral, el coeficiente de altura, el pitch, el roll y el yaw calibrados.*

Para ello se utilizó el software de optimización Solver incluido en el Excel de la suite Microsoft Office 2011 para Mac, que emplea un algoritmo conocido como Gradiente Generalizado Reducido (GRG2) [32].

La efectividad de los modelos de calibración se estudia en la figura 34. En ella se muestran una serie de gráficas en las que se representa el error absoluto de la lecturas filtradas con respecto al valor real antes (en color azul) y después (en color rojo) de aplicar los modelo de calibración para cada variable. Para cada una de las 118 posiciones (ver tabla 9) se ha calculado el valor medio de cada una de las variables (distancia lateral, frontal, etc) a partir del conjunto de 1500 lecturas devueltas por el April Tags Fiducial System en cada una de citadas posiciones. Los valores de la distancia lateral y altura calibrados se obtienen de deshacer la división inicial del siguiente modo:

$$\text{distancia lateral}_{\text{calibrada}} = \text{CL}_{calibrado}\ y_{calibrada} \quad [71]$$

$$\text{altura}_{\text{calibrada}} = \text{CA}_{calibrado}\ y_{calibrada} \quad [72]$$

Se constata que:

- Para la distancia lateral (figura 34.a) se ha logrado reducir el error máximo con respecto a la referencia real de 0.3 m a 0.21 m tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 72 de las 118 posiciones pero se empeora en las 46 restantes.

- Para la distancia frontal (figura 34.b) se ha logrado reducir el error máximo con respecto a la referencia real de 0.20 m a 0.14 m tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 98 de las 118 posiciones pero se empeora en las 20 restantes.

- Para la altura (figura 34.c) se ha logrado reducir el error máximo con respecto a la referencia real de 0.14 m a 0.10 m tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 98 de las 118 posiciones pero se empeora en las 20 restantes.

- Para el pitch (figura 34.d) se ha logrado reducir el error máximo con respecto a la referencia real de 16º a 8º tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 83 de las 118 posiciones pero se empeora en las 35 restantes.

- Para el roll (figura 34.e) se ha logrado reducir el error máximo con respecto a la referencia real de 9.6º a 7.4º tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 68 de las 118 posiciones pero se empeora en las 50 restantes.

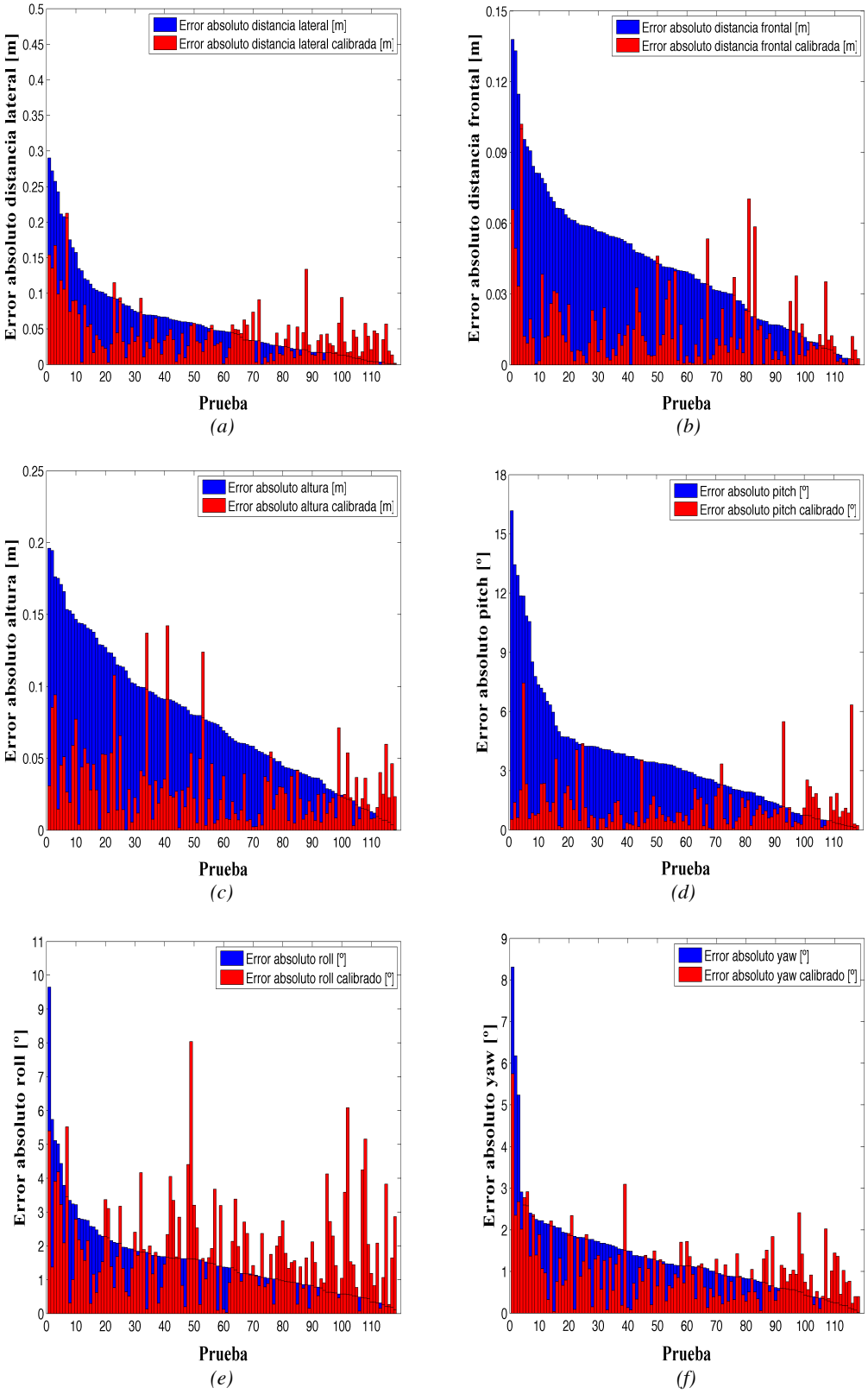


***Figura 34**. Representación del error absoluto de las lecturas del April Tags Fiducial System con respecto al valor real antes y después de la aplicación de los modelos de calibración para cada una de las 118 posiciones representadas en la tabla 9. (a) Distancia lateral. (b) Distancia frontal. (c) Altura relativa entre el tag y el AR Drone. (d) Pitch. (e) Roll. (f) Yaw.*

- Para el yaw (figura 34.f) se ha logrado reducir el error máximo con respecto a la referencia real de 8.3º a 5.7º tras la aplicación del modelo de calibración. También se observa que tras aplicarlo se mejora el error cometido en 82 de las 118 posiciones pero se empeora en las 36 restantes.

En conclusión, se observa que los modelos de calibración aplicados reducen el error máximo cometido con respecto a la medida real en todas las variables del problema (distancia lateral y frontal, altura, pitch roll y yaw). También se constata que ella diferencia entre la lectura y el valor real se reduce para todas las variables en la mayoría de las posiciones aunque con el aspecto negativo de que en un considerable número de posiciones se vea aumentado el error tras aplicar los modelos de calibración (siendo el caso más grave el del roll, que empeora en el 42 % de los casos). A pesar de ello, los modelos descritos son los que más reducen la suma del error cuadrático de todos los que se han llegado a probar, considerándose por tanto los más adecuados para calibrar las medidas.

4.4 Fusión de los datos provenientes de los sensores onboard con los provenientes del sistema April Tags.

El objetivo de este apartado es tratar el proceso de integración en el filtro de Kalman extendido de los datos procedentes de dos fuentes distintas: los sensores onboard del AR Drone y el April Tags Fiducial System. Este proceso implica en primer lugar una sincronización de los datos, ya que las fuentes que los envían no lo hacen con la misma frecuencia. Este problema fue estudiado por los investigadores de la TUM, pero se realizaron modificaciones para añadir el nuevo sistema April Tags. Esto se presenta en el apartado 4.4.1. En segundo lugar, se han desarrollado unos filtros en los que se combinan los datos procedentes de las dos fuentes con el objetivo de mejorar la precisión de las observaciones que se introducen al filtro, lo que se presenta en el apartado 4.4.2.

4.4.1 Sincronización

Un aspecto esencial cuando se quieren combinar datos de múltiples sensores, y más cuando estos se transmiten a través de una LAN inalámbrica, es la correcta sincronización de los mismos. En concreto para el AR Drone el tiempo que transcurre desde el instante en el que un frame es capturado por la cámara, y el instante en el que la señal de control es aplicada es de en torno unos 250 ms. Para analizar el proceso de comunicación con el dron (ver figura 35) hay que definir los retrasos de tiempo que nos van a afectar:

- **Retraso en las imágenes de la cámara (t_{cam})**: define el retraso que existe entre el instante en el que un frame de video es capturado y el instante en el que se decodifica y está listo para posterior procesado en el PC. Es, decir, si usamos

como referencia la figura 2, sería el tiempo asociado al envío los frames de imagen al ordenador a través de la wireless LAN más el tiempo de conversión requerido por el driver para pasar dicha información a formato ROS.

- **Retraso en los datos de altura y velocidad horizontal enviados por los sensores onboard (t_{xyz})**: define el retraso que existe entre el instante en el que el AR Drone obtiene una medición de los datos de velocidad horizontal y altura y el instante en el que estas estimaciones están disponibles en el PC. Si usamos como referencia la figura 10, sería la suma del tiempo asociado al preprocesado realizado sobre las velocidades y la altura, más el tiempo asociado al envío de estos datos a través de la wireless LAN, más del tiempo de conversión requerido por el driver para pasar dicha información a formato ROS.

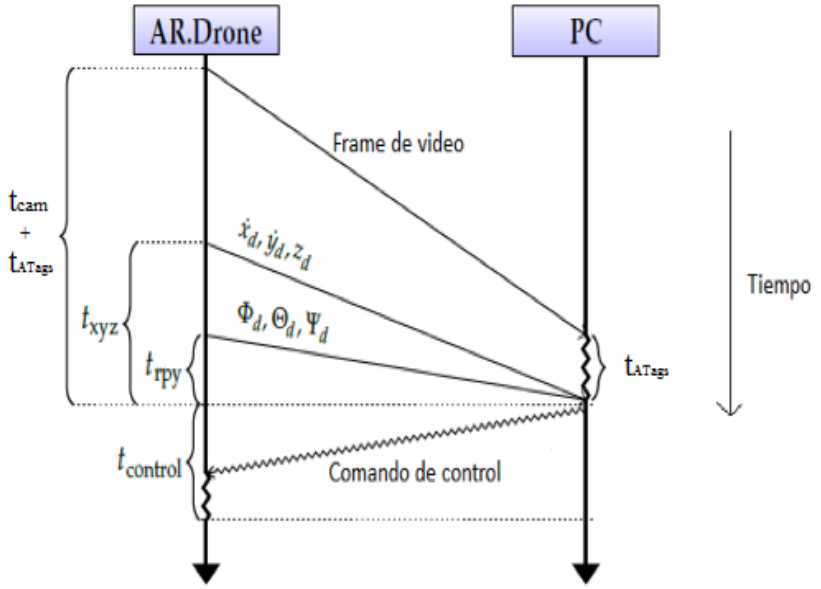


Figura 35. *Diagrama secuencial que ilustra cómo se produce la comunicación entre el AR Drone y el PC, indicándose también los respectivos retrasos definidos.*

- **Retraso en los datos de los giroscopios (t_{rpy}):** las medidas de los ángulos roll, pitch y yaw están apenas sujetas a preprocesado y por tanto son las que menor retraso tienen. Si usamos como referencia la figura 10, sería el tiempo asociado al envío de los datos medidos por la IMU que se realiza a través de la wireless LAN, más el tiempo de conversión requerido por el driver para pasar dicha información a formato ROS.

- **Retraso en el control ($t_{control}$):** cada 10 ms se calcula una señal de control actualizada en el PC, enviada al dron y usada como punto de partida para el controlador del AR Drone. El $t_{control}$ define el tiempo que transcurre desde el

instante en el que se envía el comando de control al instante en el que se aplica. Es decir, si usamos como referencia la figura 10, sería la suma del tiempo asociado al envío de los comandos de control del PID al driver, más el tiempo requerido por este para realizar la conversión de las órdenes de control a formato del AR Drone, más el tiempo de envío de estos datos a través de la wireless LAN.

- **Tiempo de procesado de imágenes ($t_{PTAM} = t_{ATags}$):** es el tiempo requerido para llevar a cabo el procesado de las imágenes en el ordenador. Si tomamos como referencia la figura 2, originalmente se relacionaba con el tiempo requerido para la aplicación de los algoritmos del PTAM y obtención de una estimación de la posición del dron. Sin embargo, como ya se ha comentado anteriormente, el sistema que se ha desarrollado en este trabajo fin de grado prescinde del uso del PTAM y lo sustituye por un sistema basado en los marcadores artificiales April Tags. Es por ello que ahora este tiempo se relaciona con el que requiere el April Tags Fiducial System para obtener la posición del tag con respecto a la cámara más el tiempo de procesado para convertir esas lecturas en una estimación de la posición del dron.

El planteamiento seguido para la compensación de los retrasos es el siguiente: primero se les asigna un sello de tiempo ("*timestamp*") a los datos entrantes (frames de video y datos procedentes de los sensores) y se almacenan en un búfer. Al AR Drone se le envían comandos de control con una frecuencia de unos 100 Hz, es decir, cada 10 ms. Sin embargo, se sabe que hay un cierto retraso para que ese comando tenga efecto ($t_{control}$). Es por esto que, para cada instante de tiempo actual t en el que enviamos un comando de control, se tiene que hacer una predicción del estado que va a tener el dron en $t + t_{control}$. Para hacer esta predicción se usa el filtro de Kalman extendido, utilizándose como punto de partida para la predicción el estado del filtro en el instante $t - t_{cam}$, es decir, el estado del filtro cuando se recibió el último frame de video. Para predecir el estado se deben integrar los datos de los sensores almacenados así como anteriores comandos de control enviados (que también se encuentran almacenados en un búfer). Hay que comentar que el filtro de Kalman se desplaza de forma permanente al instante en que se recibe un nuevo frame de video, incorporando todas las medidas de los sensores (corregidas temporalmente) como observaciones para su eliminación del búfer. Cuando el procesado realizado con el sistema April Tags se completa, la posición calculada se añade como observación al filtro, mejorando la estimación de la posición.

4.4.2 Filtrado *online*

La idea del proceso de filtrado online es obtener observaciones lo más precisas posibles mediante la combinación de los datos procedentes de las diferentes fuentes de información de las que se dispone: los sensores onboard del AR Drone y el sistema basado en los April Tags.

4.4.2.1 Filtrado de las observaciones de *x* e *y*.

En primer lugar se aplica un filtrado basado en un criterio de distancias. Se comprueba si el salto en las variables de estado *x* e *y* en 40 ms es superior a 0.2 metros. Si esto ocurre es que se ha producido algún error en la conexión inalámbrica y no se han recibido información del AR Drone durante un considerable intervalo de tiempo. Esto provoca un gran salto en la *x* e *y*, lo cual no es posible en condiciones de operación normales del dron. El diagrama de flujo del filtro para x se presenta en la figura 36.

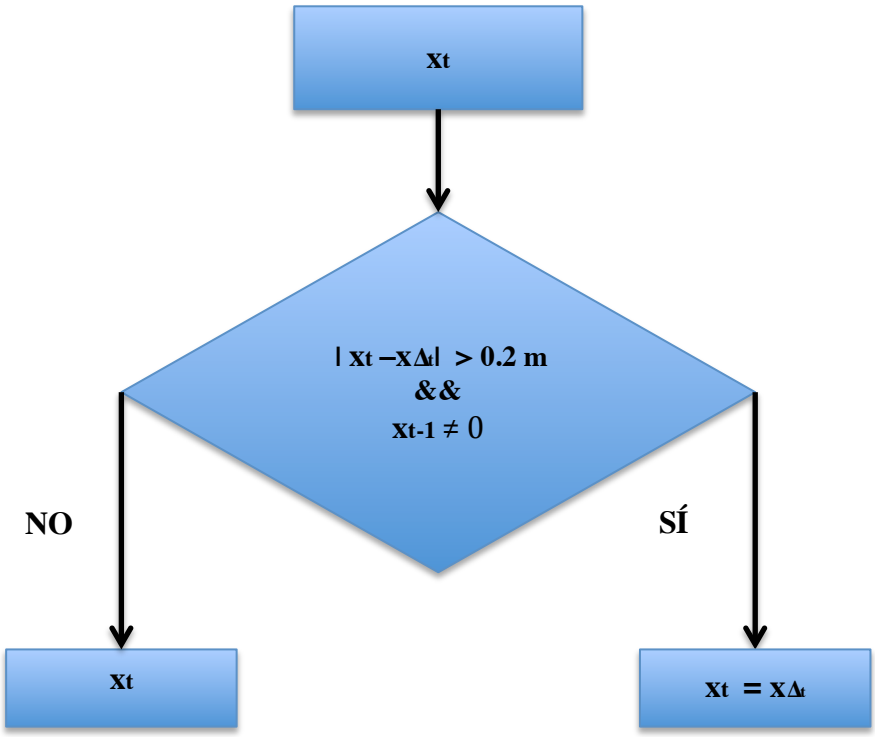


Figura 36. *Diagrama de flujo del filtro online para x. xt es el estado del filtro de Kalman en el instante de tiempo actual. xt-1 es el estado del filtro de Kalman en el instante de tiempo anterior (hace 40 ms). La variable y se procesaría de forma idéntica.*

Como ya se ha comentado, el principal problema que surge en la estimación de las variables de estado *x* e *y* cuando se navega empleando exclusivamente los sensores onboard es la deriva. La utilización de los marcadores artificiales April Tags permite solventar este problema siempre que los tengamos monitorizados, ya que nos ofrece observaciones directas y de alta precisión de las variables de estado en cuestión. Sin embargo, hay que tener en cuenta que el objetivo de este trabajo es lograr un sistema de navegación lo más general posible, siendo necesario que el AR Drone sea capaz de volar durante períodos de tiempo más o menos prolongados en los que no sea posible tener los April Tags a la vista. Durante estos períodos en los que no sea posible contar con información de los marcadores, el error en el posicionamiento irá creciendo con el tiempo a medida que el cuadricóptero se va desplazando. Cuando el periodo de tiempo sin ver tags sea prolongado, el estado del dron divergirá bastante de la posición real en

la que se encuentra, y consecuentemente, también lo hará de las lecturas de los April Tags que recibamos, ya que estas serán, muy probablemente, mucho más precisas que las del estado, que se habrá corrompido por la deriva.

Para detectar y solucionar estas situaciones, lo que se ha hecho es establecer un filtro utilizando un criterio de tiempos. Empleando una utilidad de ROS que nos permite acceder al reloj del sistema, se crea un contador del tiempo que ha pasado desde la última vez que se ha detectado un tag. Si este tiempo supera un determinado umbral (que se ha establecido de 5 segundos), consideramos que el estado ha sido corrompido por la deriva y por tanto el filtrado expuesto en la figura 36 no se aplica. Además se fuerza el reseteo del estado al valor de las lecturas que obtenemos del sistema April Tags. Una vez realizada esta reasignación, se vuelve a aplicar el criterio de la citada figura.

4.4.2.2 Filtrado de las observaciones de *z*

En este caso, los mediciones enviadas por los sensores onboard de la altura son altamente fiables por el hecho de que se trata de medidas absolutas de dicha magnitud, lo cuál elimina el problema de la deriva, y de que la optimización que se ha implementado (ver apartado 4.2.2) hace que sean altamente precisas. Es por ello que los datos procedentes de los sensores onboard se usan como referencia para filtrar las medidas procedentes del sistema April Tags. El filtro introducido se define en la figura 37.

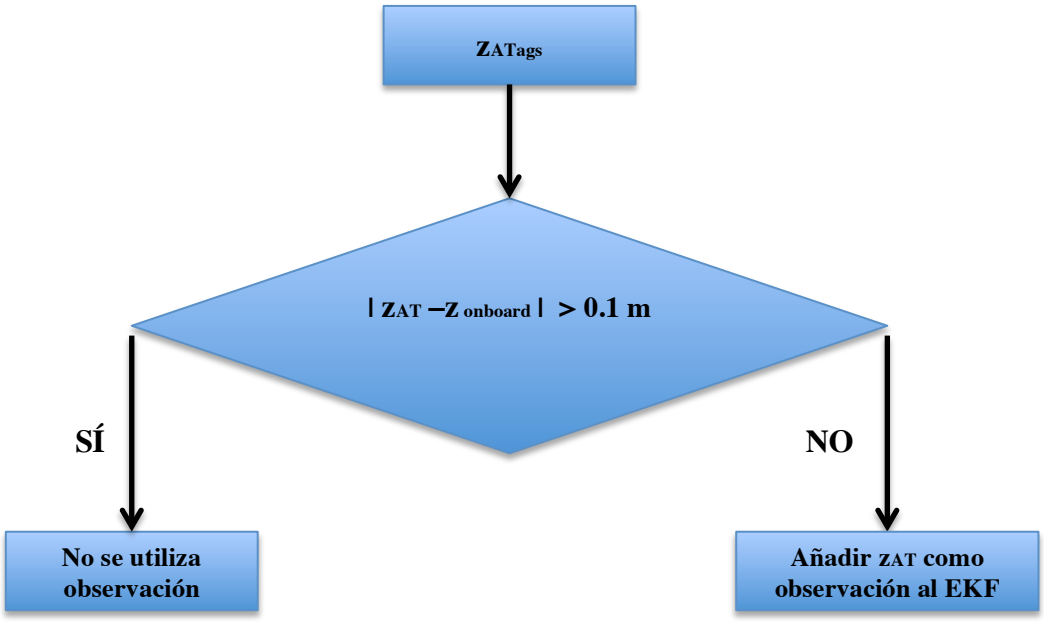


Figura 37. *Diagrama de flujo del filtro online para la z enviada por el sistema April Tags. z_{AT} es la observación de la altura del cuadricóptero calculada por el sistema April Tags para un determinado instante de tiempo. $z_{onboard}$ es la lectura de la altura del cuadricóptero obtenida tras el procesado de la información enviada por los sensores onboard del cuadricóptero para un determinado instante de tiempo.*

4.4.2.3 Filtrado de las observaciones de pitch y el roll

Las mediciones enviadas por los sensores onboard para estos ángulos son altamente fiables por el hecho de que se trata de medidas absolutas, lo cuál elimina el problema de la deriva, y de que la optimización que se ha implementado (ver apartado 4.2.3) hace que sean altamente precisas. Es por ello que los datos procedentes de los sensores onboard se usan como referencia para filtrar las medidas procedentes del sistema April Tags. El filtro introducido se define en la figura 38.

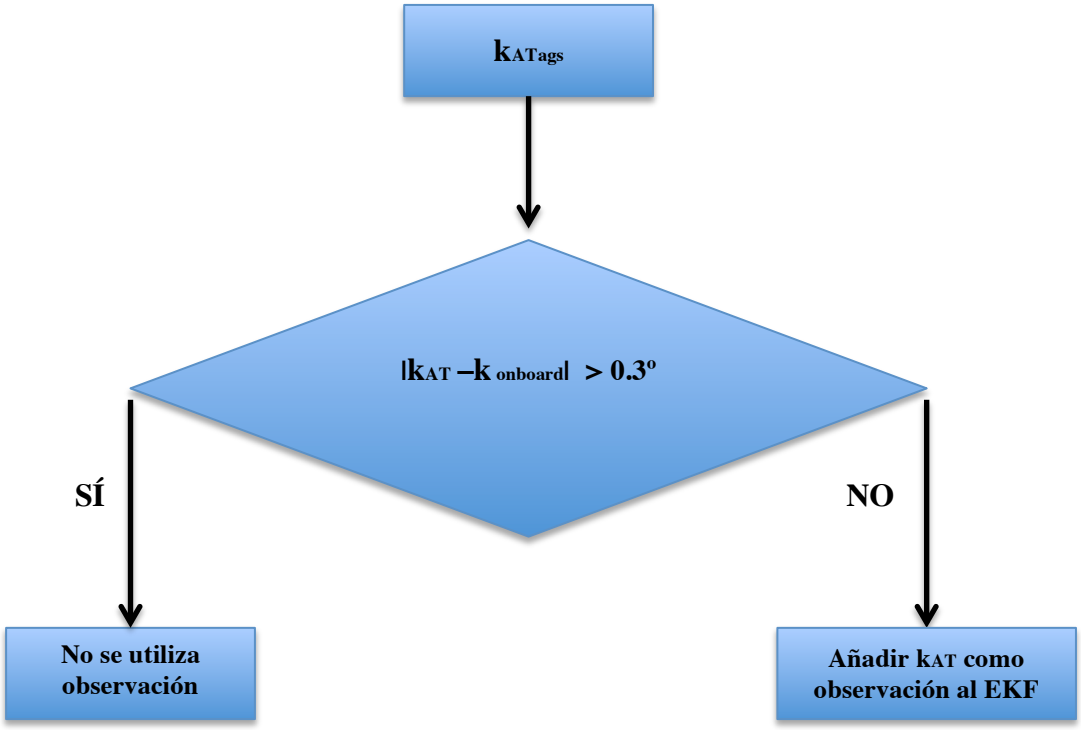


Figura 38. *Diagrama de flujo del filtro online para el pitch y el roll (simbolizados de forma genérica como k) enviados por el sistema April Tags.* k_{AT} *es la observación del ángulo pitch o roll del cuadricóptero calculado por el sistema April Tags para un determinado instante de tiempo.* $k_{onboard}$ *es la lectura del ángulo pitch o roll del cuadricóptero obtenida tras el procesado de la información enviada por los sensores onboard del cuadricóptero para un determinado instante de tiempo.*

A la hora de establecer las varianzas con las que pondera el peso de las observaciones de altura, roll y pitch, se constató en la práctica que, el incorporar las lecturas del sistema April Tags para estas magnitudes con una varianza semejante a la de los sensores Onboard, provocaba que el estado oscilase mucho más y condujese a peores estimaciones, llevando a su vez a que durante la navegación utilizando el controlador PID se tardase mucho más tiempo en considerar que se ha alcanzado un determinado punto objetivo. Es por ello que, debido a la alta precisión que ya obtenemos con de las medidas procedentes de los sensores Onboard para estas magnitudes, se decidió establecer una varianza muy grande para las lecturas de los April Tags con respecto a estos, de modo que no se tienen en consideración a la hora de estimarse el estado.

4.4.2.4 Filtrado de las observaciones del yaw

En el caso del yaw**,** como se ha comentado previamente, el principal problema que surge en su estimación cuando se navega empleando exclusivamente los sensores onboard es la deriva.

Se vio en el apartado 4.2.4 que, en vuelo, la deriva se mantenía oscilando alrededor del valor real del ángulo, creciendo la amplitud de las oscilaciones con el tiempo. El principal objetivo de añadir los April Tags es reducir esta deriva de forma que se consiga mantener la amplitud de estas oscilaciones dentro de rango razonable (máximo de $\pm 5^{o}$ en torno al valor real). Esto se consigue empleando lo mejor de las dos fuentes de información: por un lado, la información enviada por la IMU es fiable en el corto plazo, por lo que las lecturas procesadas procedentes de los sensores onboard pueden usarse para filtrar las lecturas enviadas por el sistema April Tags según el esquema que se detalla en la figura 39.

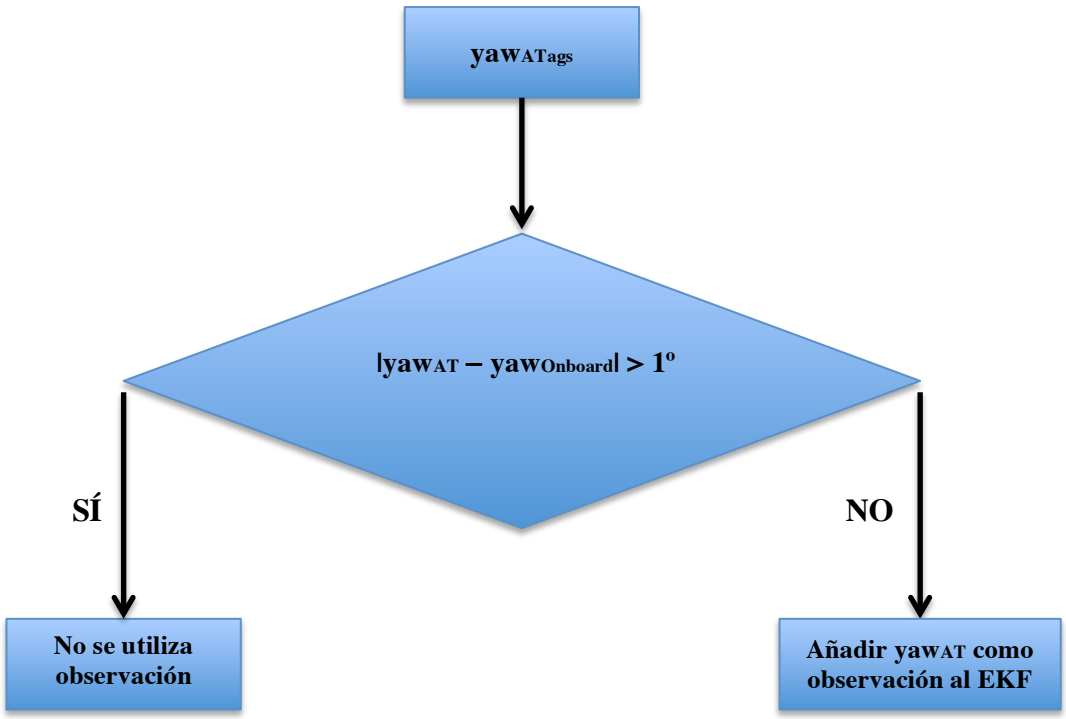


Figura 39. *Diagrama de flujo del filtro online para el yaw enviados por el sistema April Tags. yaw_{AT} es la observación del ángulo yaw del cuadricóptero calculado por el sistema April Tags para un determinado instante de tiempo. $yaw_{onboard}$ es la lectura del ángulo yaw del cuadricóptero obtenida tras el procesado de las lecturas enviadas por los sensores onboard del cuadricóptero para un determinado instante de tiempo.*

Por otro lado, el mayor peso que se le da a las medidas de los April Tags con respecto a las de la IMU a la hora de introducirlas en el filtro, hace que el estado se aproxime más a lo que dicten dichas lecturas. Esto a su vez, tiene el efecto de retroalimentación de favorecer el filtrado que se aplica en el cálculo de la observación a partir de los datos

enviados por los sensores onboard (recordar apartado 4.2.4.1), en donde se establecen como válidas únicamente las observaciones cuya diferencia con respecto al estado no sea mayor de 2º. Si esto ocurre, se omite la observación y se actualizan los baselines, con lo que evitamos que aparezca la deriva.

A pesar de todo, las tareas de navegación pueden implicar pasar un período de tiempo más o menos prolongado sin ver tags. Al igual que pasaba con las citadas magnitudes de distancia, cuando las maniobras sin tener monitorizados los tags se prolongaban demasiado, la deriva acababa corrompiendo el estado y este tomaría un valor falso que en realidad no tiene. Para detectar y solucionar estas situaciones, lo que se ha hecho es establecer un filtro utilizando un criterio de tiempos. La idea de partida es similar a lo que se aplicaba con las variables *x* e y: empleando una utilidad de ROS que nos permite medir el tiempo del sistema, se crea un contador del tiempo que ha pasado desde la última vez que se ha detectado un tag, y si este tiempo supera un determinado umbral (que se ha establecido de 5 segundos), consideramos que el estado ha sido corrompido por la deriva, en cuyo caso el filtrado expuesto en la figura 39 no se aplica.

Sin embargo, en este caso no se puede resetear directamente el estado al valor de las lecturas que obtenemos del sistema April Tags, ya que el yaw tiene una exactitud mucho menor que las lecturas de las variables *x* e *y*. Es por ello que si se recupera la monitorización de los tags tras un periodo prolongado de tiempo, se crea un vector con las primeras 7 observaciones de yaw que se obtienen del sistema April Tags y se calcula su desviación típica. Si dicha desviación es mayor de 2 grados se repite el proceso con las 7 medidas siguientes hasta que se cumpla el citado criterio de desviación, lo que querrá decir que las lecturas que se obtienen del sistema April Tags son estables. El valor obtenido se considera mucho más preciso que el del estado, por lo que se procede a resetear el mismo al valor de la media del vector (ver figura 40). Una vez se añade la observación se vuelve a aplicar el filtrado expuesto en la figura 39.

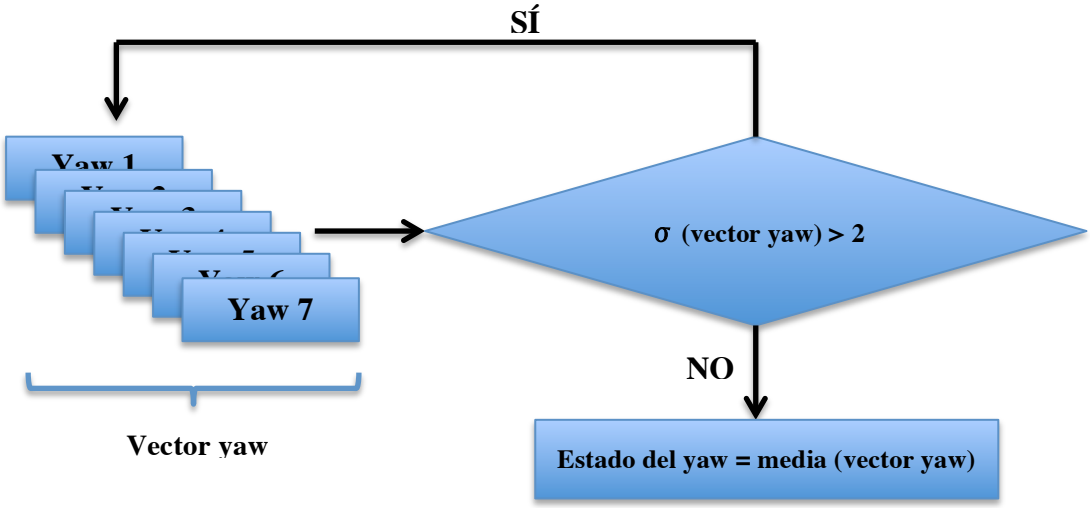


Figura 40. *Esquema de funcionamiento del filtro que se aplica al yaw cuando se recupera la monitorización de los April Tags.*

5. Pruebas de validación del sistema de navegación.

El objetivo del presente capítulo es presentar las pruebas que se llevaron a cabo para comprobar el correcto funcionamiento del sistema diseñado. En primer lugar se evalúa la capacidad del cuadricóptero de realizar una determinada trayectoria en presencia de un obstáculo. En segundo lugar se evalúa la capacidad del cuadricóptero de realizar múltiples giros y volver al punto de partida con una orientación similar a la inicial, teniéndose monitorizados en todo momento los April Tags. Por último, se busca evaluar la capacidad del cuadricóptero de reubicarse tras pasar un periodo prolongado de tiempo navegando sin tener monitorizados los April Tags.

5.1 Prueba 1: vuelo con obstáculos.

El experimento se ha llevado a cabo en un entorno como el que se ilustra en la figura 41. El origen del sistema mundo S_0 (ver figura 28) se ha definido de tal forma que el marcador April Tag id 2 se encuentra en la posición x = 0 m, y = 2.462 m , z = 0.69 m, el tag id 4 se encuentra a unas coordenadas de x = -1.206 m, y = 2.462 m , z = 0.69 m y el id 3 a x = 1.206 m, y = 2.462 m , z = 0.69 m. Se ha empleado un suelo tipo 1 para la realización de la maniobra con el fin de asegurar que la estimación de las velocidades por aplicación de los algoritmos de flujo óptico sea la óptima.

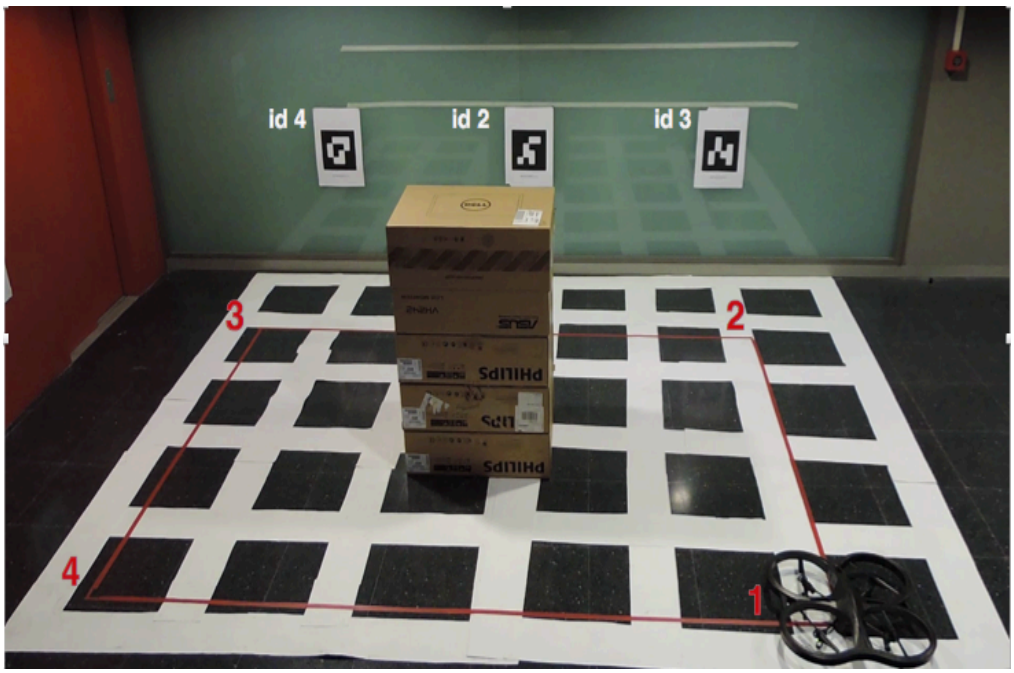


***Figura 41**. Montaje experimental de la prueba de validación 1, consistente en dar dos vueltas alrededor de la caja de 1 m de altura que se observa en la imagen. La trayectoria seguida por el cuadricóptero se ha marcado en el suelo mediante cinta de color rojo.*

Se han dispuesto una serie de cajas creando un obstáculo monolítico de 1 metro de altura que será bordeado por el cuadricóptero dando dos vueltas a su alrededor según una trayectoria cuadrada como la que se observa en la figura 41, trazada en el suelo del

laboratorio con cinta de color rojo. Las coordenadas de los vértices del cuadrado con respecto al sistema de coordenadas S_0 son *(x,y) = (1.3, -0.5)* para el punto 1, *(x,y) = (1.3, 1.3)* para el punto 2, *(x,y) = (-1.3, 1.3)* para el punto 3, y *(x,y) = (-1.3, -0.5)* para el punto 4. La maniobra se ha programado con el controlador de alto nivel según la ruta que se muestra en la tabla 11.

Punto	Target x [m]	Target y [m]	Target z [m]	Target yaw [º]
2	1.3	1.3	0.7	0
	0	1.3	0.7	0
3	-1.3	1.3	0.7	0
	-1.3	0	0.7	0
4	-1.3	-0.5	0.7	0
	0	-0.5	0.7	0
1	1.3	-0.5	0.7	0

***Tabla 11**. Resumen de la ruta de puntos objetivo establecidos en el controlador de alto nivel que se envían al PID para que los ejecute.*

Dichos puntos objetivos se envían al PID que los interpreta y los convierte en comandos de velocidad que, a su vez, son enviados al AR Drone. Recuérdese que la ruta señalada en la tabla 11 se repite dos veces, ya que el cuadricóptero da dos vueltas alrededor del obstáculo. Obviamente, aunque no aparezcan en la tabla, también se le envían al AR Drone las órdenes de despegue y aterrizaje. En la figura 42 se han representado las variables de estado x (figura 42.a), y (figura 42.b), z (figura 42.c), pitch (figura 42.d), roll (figura 42.e) y yaw (figura 42.f) en color azul, así como las observaciones obtenidas con el sistema April Tags para (x,y,Ψ) en color morado frente al tiempo. En las figura 43 se muestran una serie de capturas de imagen de los puntos más relevantes de la maniobra. Dichas capturas se han tomado de una grabación de video del vuelo realizado, por lo que representan la posición del dron en la realidad. Si comparamos las figuras 42 y 43 observamos que:

- La figura 43.a se corresponde con el target $(x,y,z,\Psi)=(1.3,1.3,0.7,0)$, que equivale al punto 2 de la trayectoria cuadrada señalada en la figura 41. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a 1.3 m para x e y, así como en torno a 0.7 m para z, lo cual se ajusta mucho a la posición real.

- La figura 43.b se corresponde con el target $(x,y,z,\Psi)=(-1.3,1.3,0.7,0)$, que equivale al punto 3 de la trayectoria cuadrada señalada en la figura 41. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a -1.3 m para la x, 1.3 m para la y, así como 0.7 m para z, lo cual se aproxima bastante a la posición real.

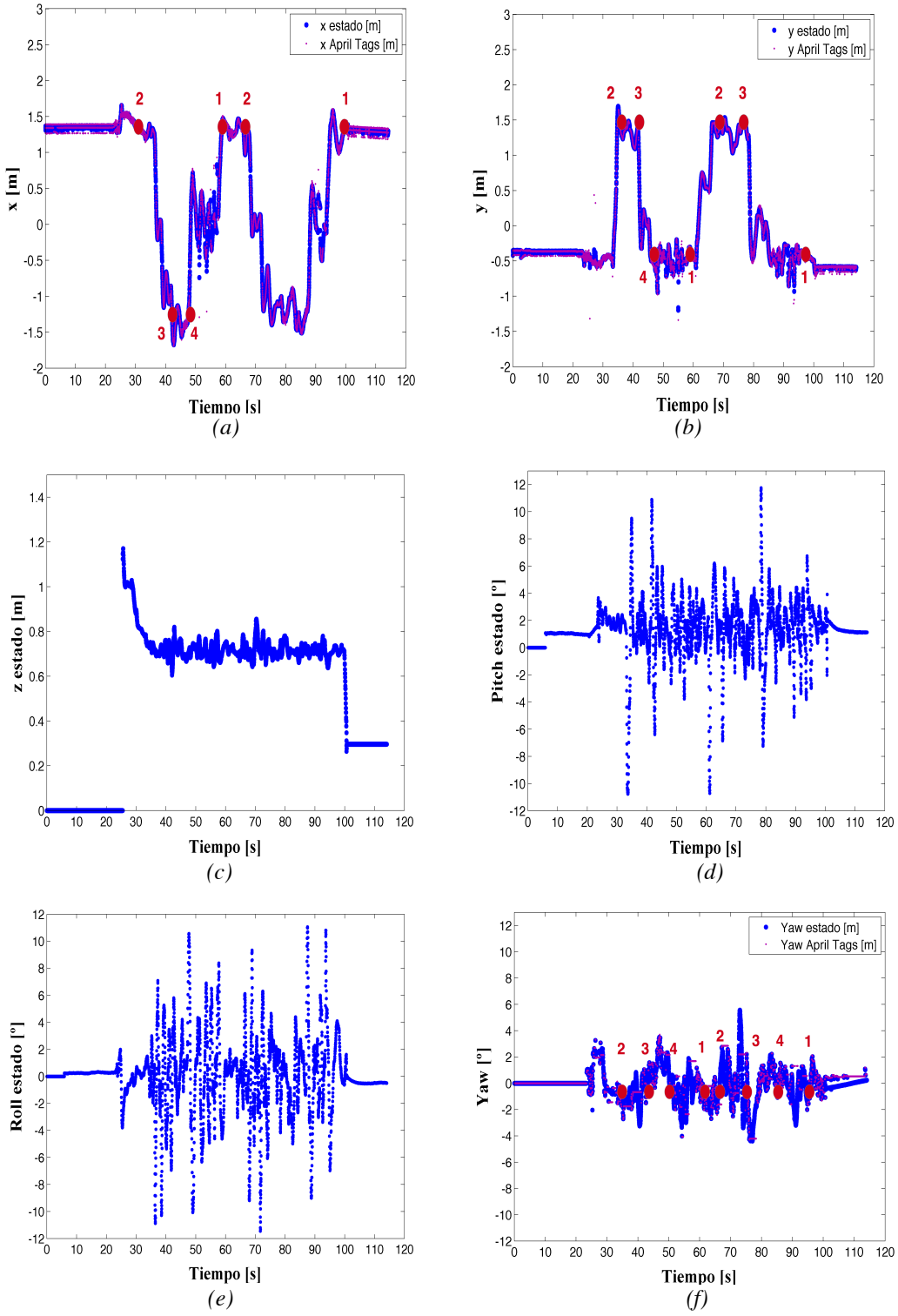
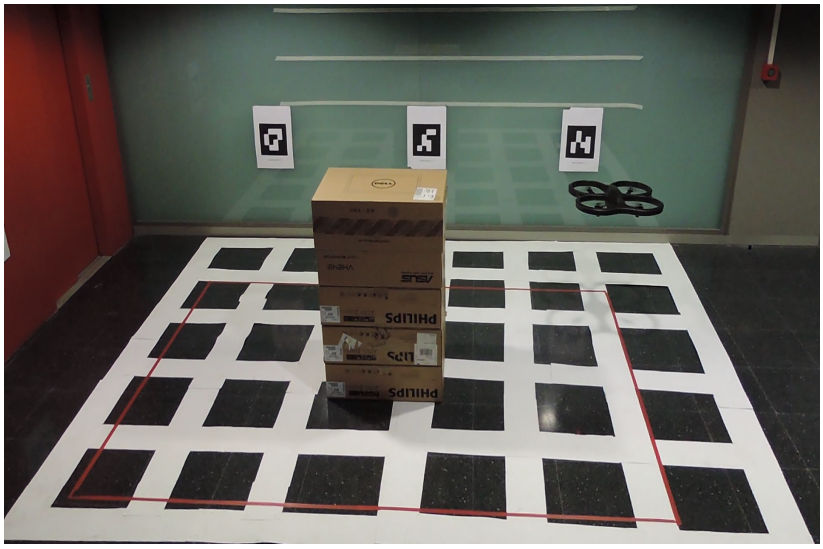
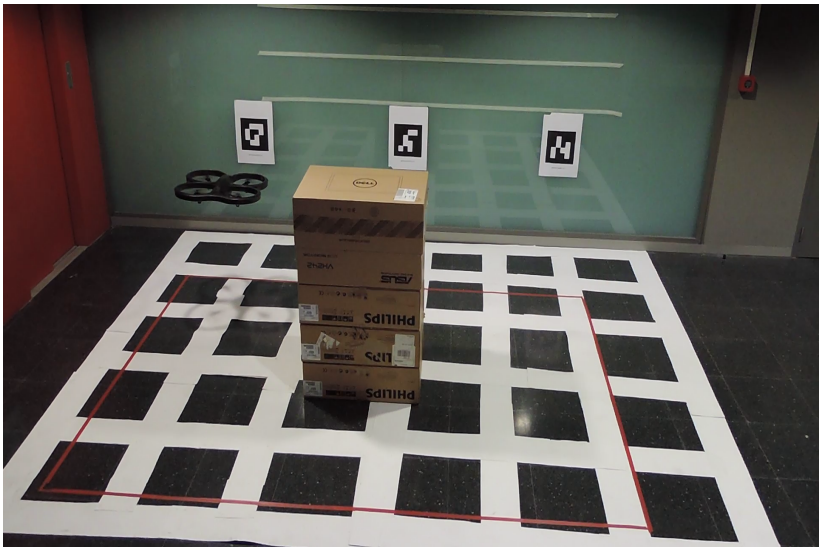


Figura 42. *Representación de las variables de estado del cuadricóptero (en color azul) y de las lecturas del sistema April Tags (en morado) frente al tiempo para la maniobra que se detalla en la tabla 11. (a) Coordenada x. (b) Coordenada y. (c) Coordenada z. (d) Ángulo pitch. (e) Ángulo roll. (f) Ángulo yaw. Las medidas de los April Tags solo se observan en (a), (b) y (f) porque para el resto de variables se han introducido con varianzas muy elevadas que hacen que el filtro no las considere. Se han señalado mediante marcadores circulares de color rojo los instantes en los que el cuadricóptero se encuentra sobre los vértices de la trayectoria (ver figura 41).*

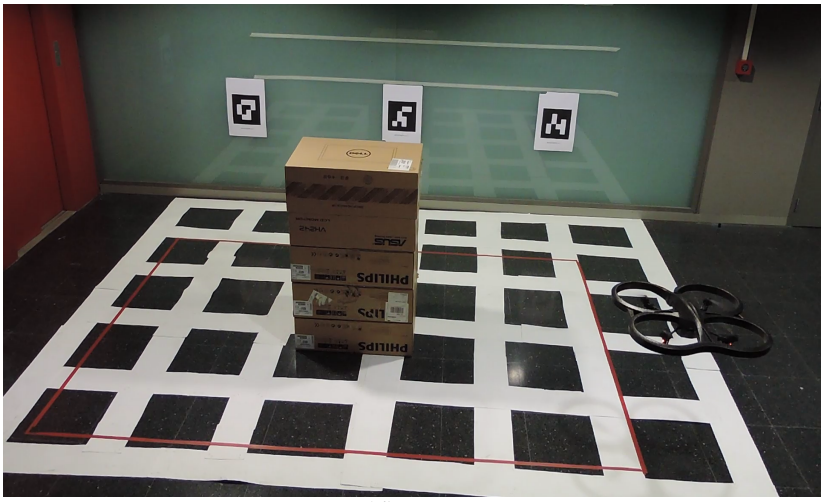


(a)

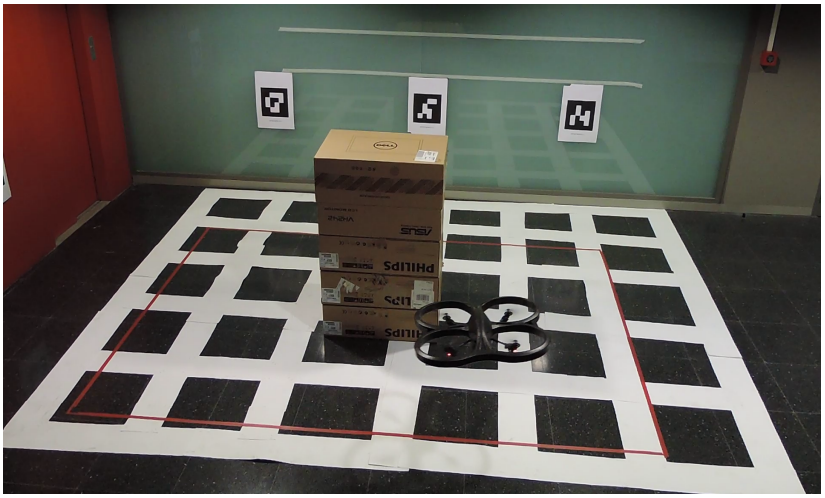


(b)

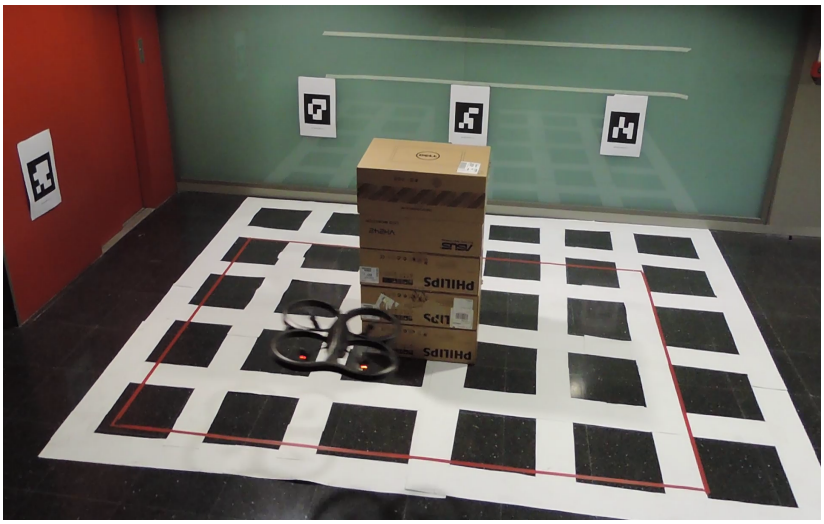
(c)



(d)



(e)



(f)

Figura 43. *Capturas de video que representan diversos puntos de la maniobra de vuelo que se le ha ordenado realizar al AR Drone. (a) Coordenadas (x,y,z,Ψ) = (1.3,1.3,0.7,0). (b) Coordenadas (x,y,z,Ψ) = (-1.3,1.3,0.7,0). (c) Coordenadas (x,y,z,Ψ) = (-1.3,-0.5,0.7,0). (d) Coordenadas (x,y,z,Ψ) = (1.3,-0.5,0.7,0). (e) y (f) Coordenadas (x,y,z,Ψ) = (0,-0.5,0.7,0).*

- La figura 43.c se corresponde con el target $(x,y,z,\Psi)=(-1.3,-0.5,0.7,0)$, que equivale al punto 4 de la trayectoria cuadrada señalada en la figura 41. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a -1.3 m para la *x*, -0.5 m para la *y*, así como 0.7 m para *z*, lo cual se aproxima bastante a la posición real.

- La figura 43.d se corresponde con el target $(x,y,z,\Psi)=(1.3,-0.5,0.7,0)$ que equivale de nuevo al punto 1 de la trayectoria cuadrada señalada en la figura 41. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a 1.3 m para la *x*, -0.5 m para la *y*, así como 0.7 m para *z*, lo cual se aproxima bastante a la posición real.

- Las figuras 43.e y 43.f se corresponden con el target $(x,y,z)=(0,-0.5,0.7)$. Se observa que las lecturas de estado para *x* e y oscilan mucho en torno al valor real (entre -0.7 m y 0.7 m aproximadamente para la primera variable y entre -0.8 m y -0.2 m para la segunda). Estas oscilaciones se observan también en la realidad, especialmente durante la primera vuelta, y se deben a que los April Tags no se mantienen constantemente monitorizados por la cámara, sino que durante breves intervalos de tiempo no se ven y eso hace que el posicionamiento del dron sea más impreciso y tarde más en encontrar la posición objetivo que en el resto de los casos. En lo que respecta a la variable *z*, se mantiene en torno a 0.7 lo cual se aproxima bastante a la posición real.

- En cuanto al roll y el pitch (figuras 43.d y 43.e respectivamente), se observa que oscila entre valores angulares entre los -10 y 10º. Esto podría llamar la atención, pero se explica porque el dron realiza desplazamientos bastante bruscos en *x* e *y* lo cual implica una cierta inclinación del dron en su desplazamiento.

- Por último, el yaw (figura 43.f) se mantiene dentro de los límites esperados de $\pm 5^{\circ}$ en torno al valor angular real (0º). Se constata que las lecturas de los April Tags son sometidas al filtrado porque son valores mucho más puntuales que el continuo que se puede observar en las gráficas para *x* e *y*.

5.2 Prueba 2: vuelo con múltiples cambios de orientación.

Este segundo experimento se ha llevado a cabo en un entorno como el que se ilustra en la figura 44. El origen del sistema mundo S_0 (ver figura 28) se ha definido de tal forma que el marcador April Tag id 2 se encuentra en la posición *x* = 0 m, *y* = 2.462 m , *z* = 0.69 m, el tag id 4 se encuentra a unas coordenadas de *x* = -1.206 m, *y* = 2.462 m , *z* = 0.69 m, el id 3 a *x* = 1.206 m, *y* = 2.462 m , *z* = 0.69 m y el id 5 en la posición *x* = -

2.462 m, y = 0.804 m , z = 0.69 m. Se ha empleado un suelo tipo 1 para la realización de la maniobra con el fin de asegurar que la estimación de las velocidades por aplicación de los algoritmos de flujo óptico sea la óptima.

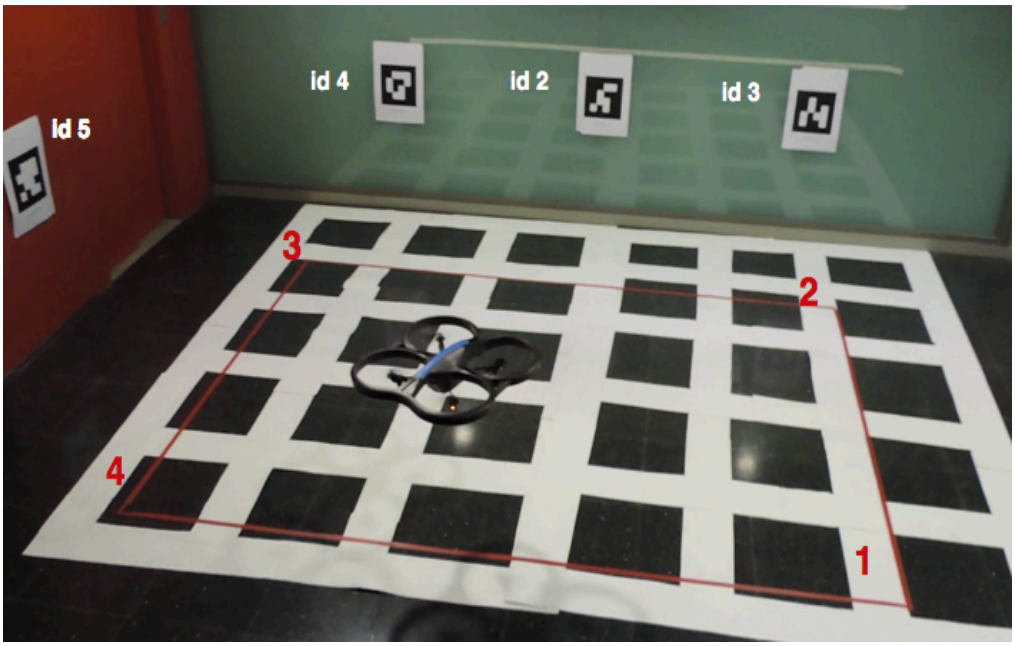


***Figura 44**. Montaje experimental de la prueba de validación 2, consistente describir la trayectoria que se ha marcado en el suelo mediante cinta de color rojo, realizando cambios de orientación en varios puntos de la misma.*

La trayectoria ha seguir por el cuadricóptero se ha trazado en el suelo del laboratorio con cinta de color rojo. También se han colocado una cintas de color azul sobre el AR Drone para dar cuenta de la orientación del mismo, ya que en esta prueba se van a realizar múltiples giros e interesa tener una referencia real de cómo de bien los realiza. Las coordenadas de los vértices del cuadrado con respecto al sistema de coordenadas S_0 son *(x,y) = (1.3,-0.5)* para el punto 1, *(x,y) = (1.3,1.3)* para el punto 2, *(x,y) = (-1.3,1.3)* para el punto 3, y *(x,y) = (-1.3,-0.5)* para el punto 4.

La maniobra se ha programado con el controlador de alto nivel según la ruta que figura en la tabla 12. Dichos puntos objetivos se envían al PID que los interpreta y los convierte en comandos de velocidad que envía al AR Drone. Obviamente también se le envían desde el controlador las órdenes de despegue y aterrizaje. En la figura 45 se han representado las variables de estado del cuadricóptero x (figura 45.a), y (figura 45.b), z (figura 45.c), pitch (figura 45.d), roll (figura 45.e) y yaw (figura 45.f) en color azul, así como las observaciones obtenidas con el sistema April Tags para (x,y,Ψ) en color morado frente al tiempo.

Punto	Target x [m]	Target y [m]	Target z [m]	Target yaw [º]
2	1.3	1.3	0.7	0
2’	1.3	1.3	0.7	-90
	0	1.3	0.7	-90
	0	1.3	0.7	0
3	-1.3	1.3	0.7	0
	-1.3	0	0.7	0
4	-1.3	-0.5	0.7	0
4’	-1.3	-0.5	0.7	30
	0	-0.5	0.7	30
	0	-0.5	0.7	0
1	1.3	-0.5	0.7	0
1’	1.3	-0.5	0.7	-30

***Tabla 12**. Resumen de la ruta de puntos objetivo establecidos en el controlador de alto nivel que se envían al PID para que los ejecute. Se han utilizado comillas para diferenciar las posiciones que coinciden en coordenadas x, y, z pero que tienen un yaw diferente.*

En la figura 46 se muestran una serie de capturas de imagen de los puntos más relevantes de la maniobra. Dichas capturas se han tomado de una grabación de video del vuelo realizado, por lo que representan la posición del dron en la realidad. Si comparamos las figuras 45 y 46 observamos que:

- La figura 46.a se corresponde con el punto 2 señalado en la figura 44, es decir, con el target $(x,y,z,\Psi)=(1.3,1.3,0.7,0)$. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a 1.3 m para la x, 1.3 m para la y, así como 0.7 m para z. En lo que respecta al yaw se constata que está muy próximo a cero. Todo ello se corresponde con lo que se observa en la realidad.

- La figura 46.b se y corresponde con el punto 2’, es decir, con el target $(x,y,z,\Psi)=(1.3,1.3,0.7,-90)$. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a 1.3 m para la x, 1.3 m para la y, así como 0.7 m para z. En lo que respecta al yaw se constata que se ha desplazado a 90º y permanece cerca de este valor. Todo ello se corresponde con lo que se observa en la realidad.

- La figura 46.d se corresponde con el punto 3, es decir, con el target $(x,y,z,\Psi)=(-1.3,1.3,0.7,0)$. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a -1 m para la x, 1.3 m para la y, así como 0.7 m para z. En lo que respecta al yaw se constata que está muy próximo a cero. Todo ello se corresponde con lo que se observa en la realidad.

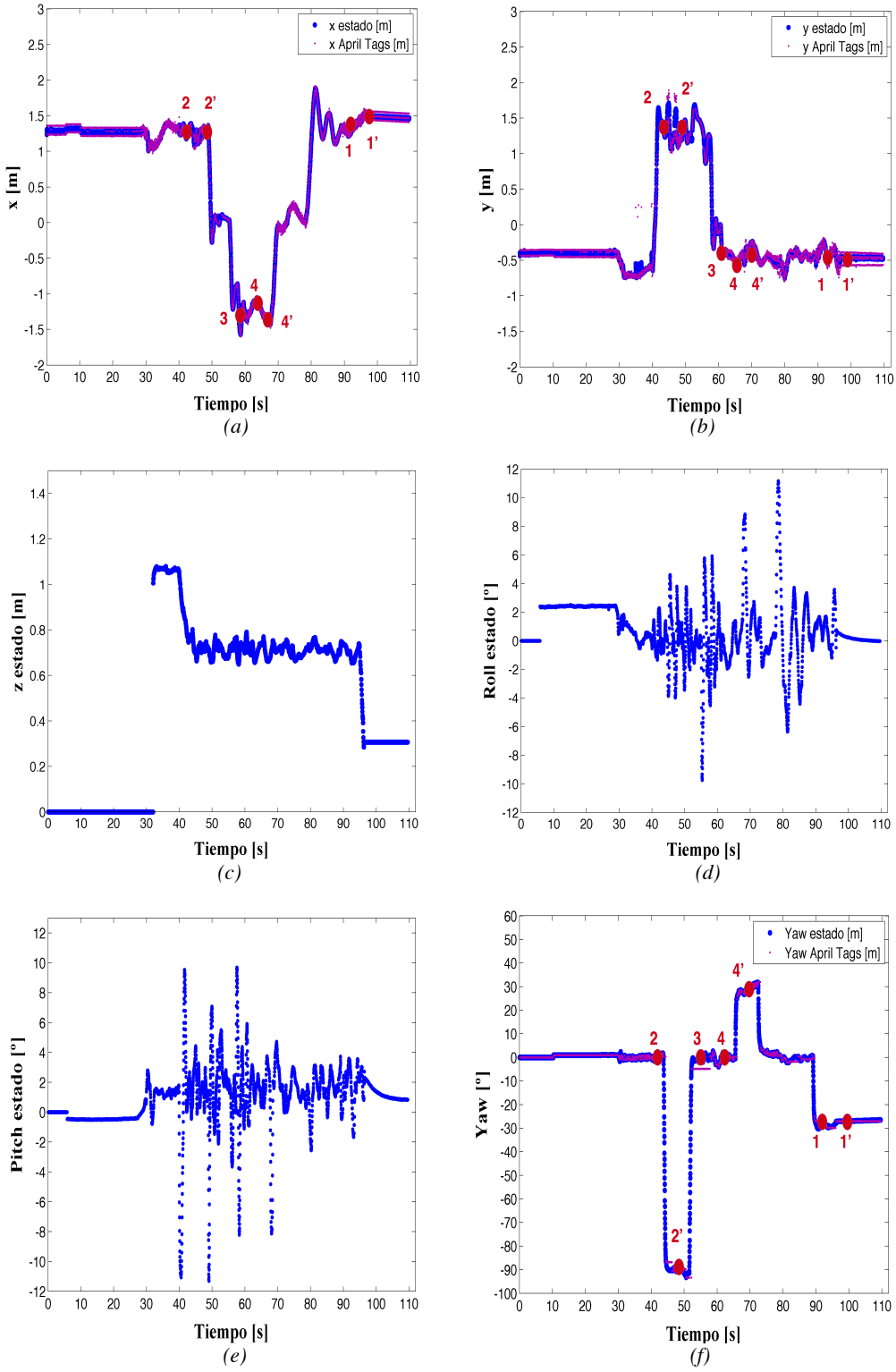
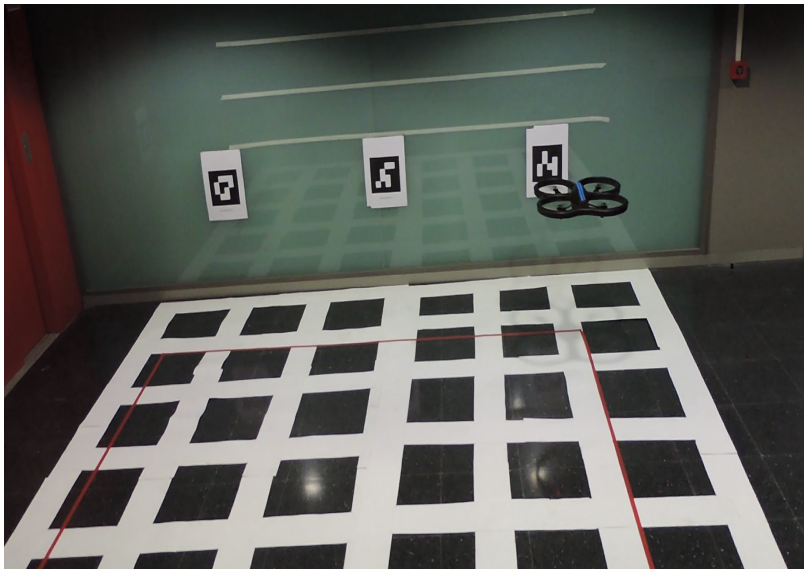
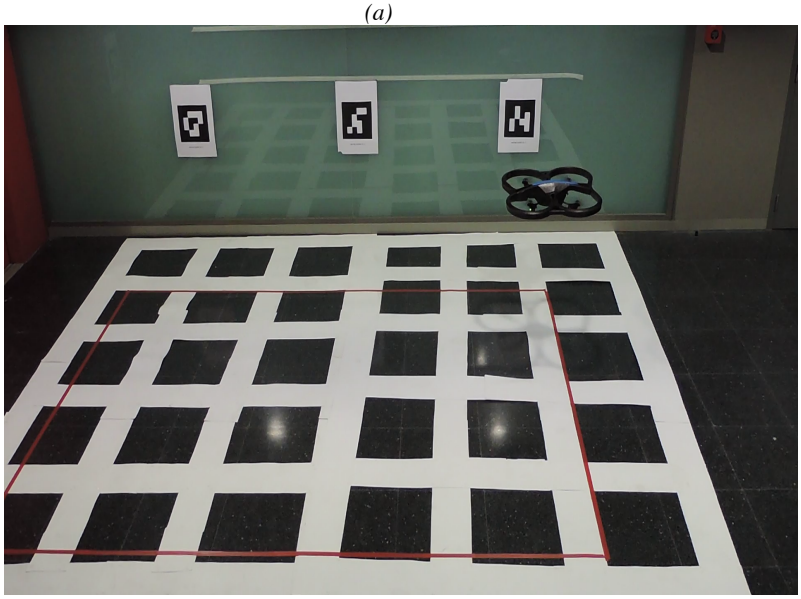


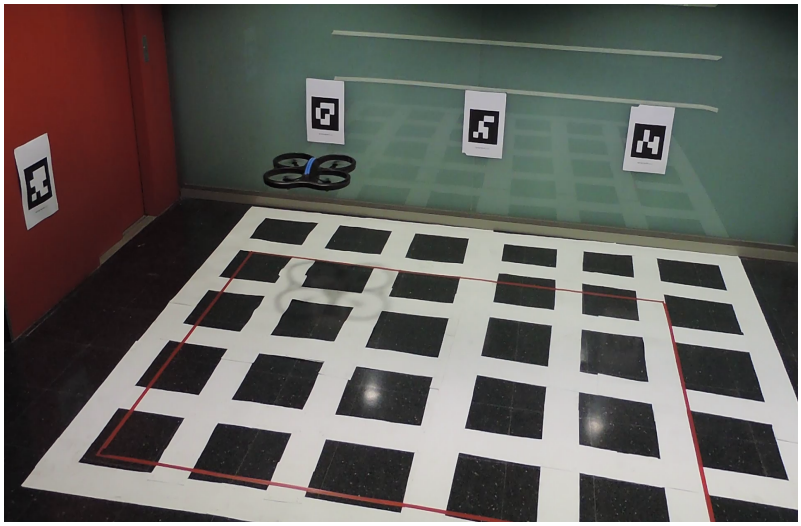
Figura 45. *Representación de las variables de estado del cuadricóptero (en color azul) y de las lecturas del sistema April Tags (en morado) frente al tiempo para la maniobra que se detalla en la tabla 12. (a) Coordenada x. (b) Coordenada y. (c) Coordenada z. (d) Ángulo pitch. (e) Ángulo roll. (f) Ángulo yaw. Las medidas de los April Tags solo se observan en (a), (b) y (f) porque para el resto de variables se han introducido con varianzas muy elevadas que hacen que el filtro no las considere. Se han señalado mediante marcadores circulares de color rojo los instantes en los que el cuadricóptero se encuentra sobre los vértices de la trayectoria (ver figura 44).*



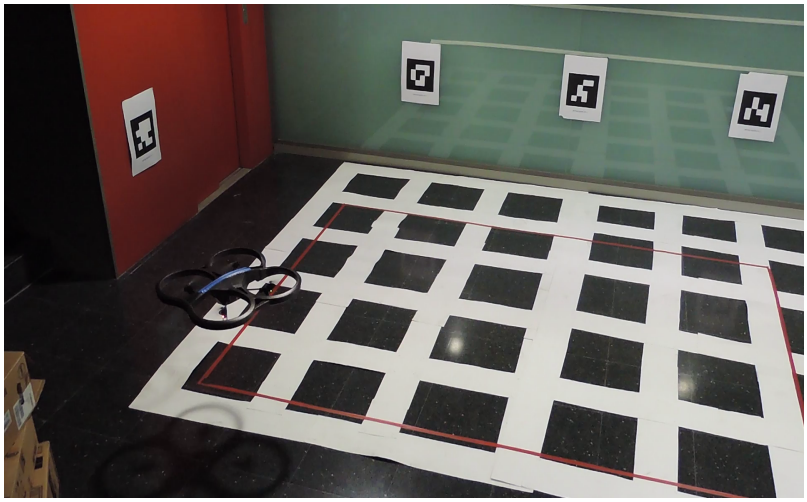
(a)



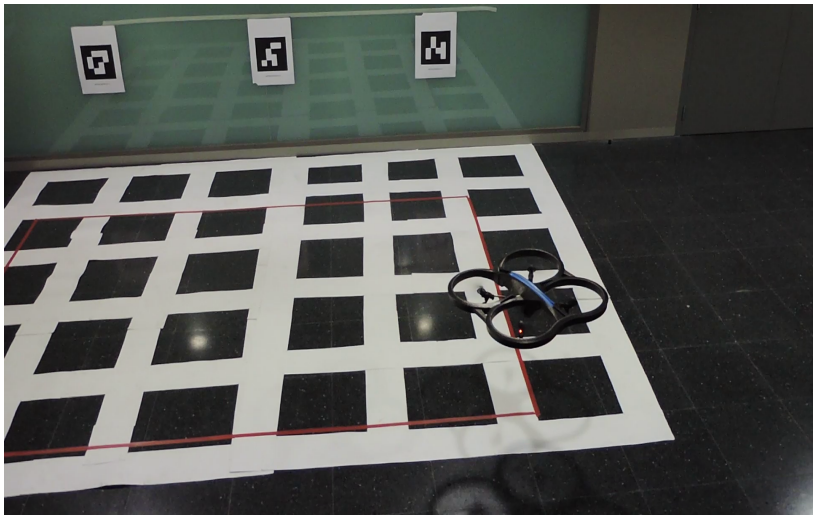
(b)



(c)



(d)



(e)

***Figura 46**. Capturas de video que representan diversos puntos de la maniobra de vuelo. (a) Coordenadas (x,y,z,Ψ) = (1.3,1.3,0.7,0). (b) Coordenadas (x,y,z,Ψ) = (1.3,1.3,0.7,-90). (c) Coordenadas (x,y,z,Ψ) = (-1.3,1.3,0.7,0). (d) Coordenadas (x,y,z,Ψ) = (-1.3,-0.5,0.7,30). (e) Coordenadas (x,y,z,Ψ) = (1.3,-0.5,0.7,-30).*

- La figura 46.e se corresponde con punto 4', es decir, con el target $(x,y,z,\Psi)=(-1.3,-0.5,0.7,30)$. Se observa que las lecturas de estado para ese punto se encuentran oscilando en torno a -1.3 m para la x, -0.5 m para la y, así como 0.7 m para z. En lo que respecta al yaw se constata que está muy próximo a 30º. Todo ello se corresponde con lo que se observa en la realidad.

- La figura 46.f se corresponde con el punto 1', es decir, con el target $(x,y,z,\Psi)=(1.3,-0.5,0.7,-30)$. Se observa que las lecturas de estado para ese punto se encuentran en torno a los 1.4 m para la x (algo por encima del valor real), oscilando alrededor de -0.5 m para la y, así como 0.7 m para z. En lo que respecta al yaw se constata que está muy próximo a -30º. Todo ello se corresponde con lo que se observa en la realidad.

- En cuanto al roll y el pitch (figuras 46.d y 46.e respectivamente), se observa que oscila entre valores angulares entre los -10 y 10º. Esto podría llamar la atención, pero se explica porque el dron realiza desplazamientos bastante bruscos en x e y lo cual implica una cierta inclinación del dron en su desplazamiento.

5.3 Prueba 3: vuelo con pérdida de monitorización de marcadores April Tags.

Este tercer experimento se ha llevado a cabo en un entorno como el que se ilustra en la figura 47. El origen del sistema mundo S_0 (ver figura 28) se ha definido de tal forma que el marcador April Tag id 2 se encuentra en la posición $x = 0$ m, $y = 2.462$ m , $z = 0.69$ m, el tag id 4 se encuentra a unas coordenadas de $x = -1.206$ m, $y = 2.462$ m , $z = 0.69$ m y el id 3 a $x = 1.206$ m, $y = 2.462$ m , $z = 0.69$ m. Se ha empleado un suelo tipo 1 para la realización de la maniobra con el fin de asegurar que la estimación de las velocidades por aplicación de los algoritmos de flujo óptico sea la óptima.

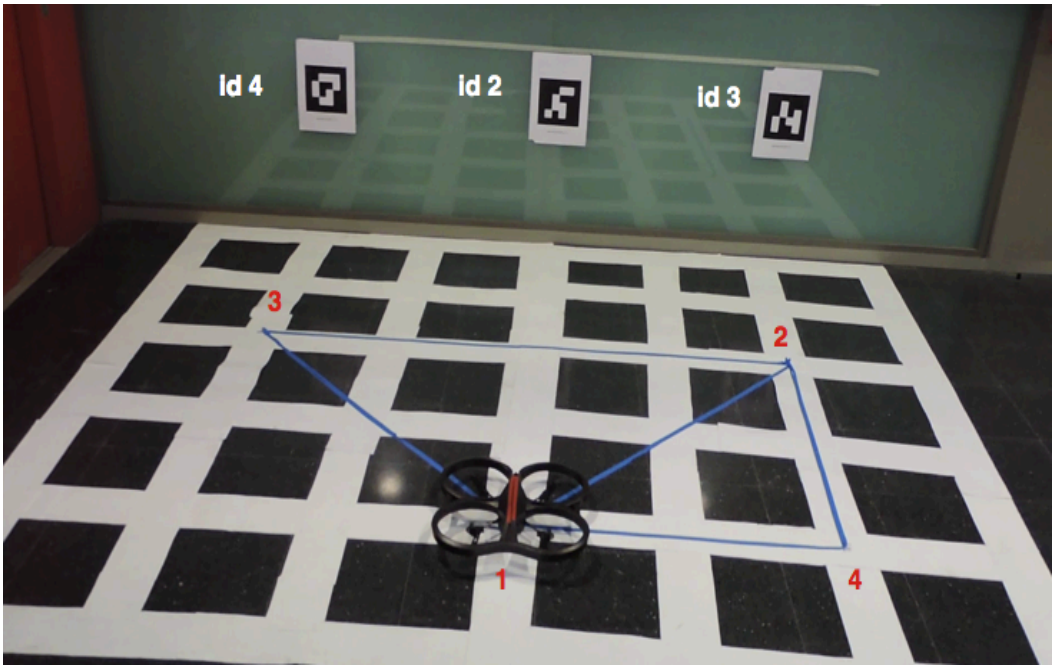


Figura 47. *Montaje experimental de la prueba de validación 3. La trayectoria a seguir consiste en realizar el triángulo 123 teniendo monitorizados los tags, a continuación volver al punto 1, girar 90º y describir dos veces la trayectoria 124, volver a 1, girar -90º para recuperar la orientación 0º y describir el triángulo 123. Por último se vuelve a 1 y se aterriza.*

La trayectoria ha seguir por el cuadricóptero se ha trazado en el suelo del laboratorio con cinta de color azul. También se han colocado una cintas de color rojo sobre el AR Drone para dar cuenta de la orientación del mismo, ya que en esta prueba se van a realizar cambios de orientación e interesa tener una referencia real de cómo de bien los realiza. Las coordenadas de los vértices de la trayectoria con respecto al sistema de coordenadas S_0 son $(x,y) = (0,0)$ para el punto 1, $(x,y) = (1.2,1.2)$ para el punto 2,

$(x,y)=(-1.2,1.2)$ para el punto 3, y $(x,y)=(1.2,0)$ para el punto 4. La maniobra se ha programado con el controlador de alto nivel según la ruta que figura en la tabla 13. Dichos puntos objetivos se envían al PID que los interpreta y los convierte en comandos de velocidad que envía al AR Drone. Obviamente también se le envían desde el controlador las órdenes de despegue y aterrizaje.

Punto	Target x [m]	Target y [m]	Target z [m]	Target yaw [º]
1	0	0	0.7	0
2	1.2	1.2	0.7	0
	0	1.2	0.7	0
3	-1.2	1.2	0.7	0
1	0	0	0.7	0
1'	**0**	**0**	**0.7**	**90**
2'	**1.2**	**1.2**	**0.7**	**90**
4	**1.2**	**0**	**0.7**	**90**
1'	**0**	**0**	**0.7**	**90**
2'	**1.2**	**1.2**	**0.7**	**90**
4	**1.2**	**0**	**0.7**	**90**
1	0	0	0.7	90
1	0	0	0.7	0
2	1.2	1.2	0.7	0
	0	1.2	0.7	0
3	-1.2	1.2	0.7	0
1	0	0	0.7	0

***Tabla 13**. Resumen de la ruta de puntos objetivo establecidos en el controlador de alto nivel que se envían al PID para que los ejecute. Se han destacado en negrita los puntos en los que el cuadricóptero no tiene monitorizados los April Tags. Se han utilizado comillas para diferenciar las posiciones que coinciden en coordenadas x, y, z pero que tienen un yaw diferente.*

En la figura 48 se han representado las variables de estado del cuadricóptero *x* (figura 48.a), *y* (figura 48.b), *z* (figura 48.c), pitch (figura 48.d), roll (figura 48.e) y yaw (figura 48.f) en color azul, así como las observaciones obtenidas con el sistema April Tags para (x,y,Ψ) en color morado frente al tiempo. En la figura 49 se muestran una serie de capturas de imagen de los puntos más relevantes de la maniobra. Dichas capturas se han tomado de una grabación de video del vuelo realizado, por lo que representan la posición del dron en la realidad. Si comparamos las figuras 48 y 49 observamos que:

- La figuras 49.a y 49.b se corresponden con los target $(x,y,z,\Psi)=(1.2,1.2,0.7,0)$ y $(x,y,z,\Psi)=(-1.2,1.2,0.7,0)$. Estos conforman dos de los vértices (puntos 2 y 3) de la primera trayectoria triangular que describe el cuadricóptero teniendo los tags monitorizados. Se observa que para el punto 2 las lecturas de estado alcanzan para la variable *x* el valor de 1.2 m, mientras que para la *y* se mantiene en un valor algo menor, en torno a 1.1 m.

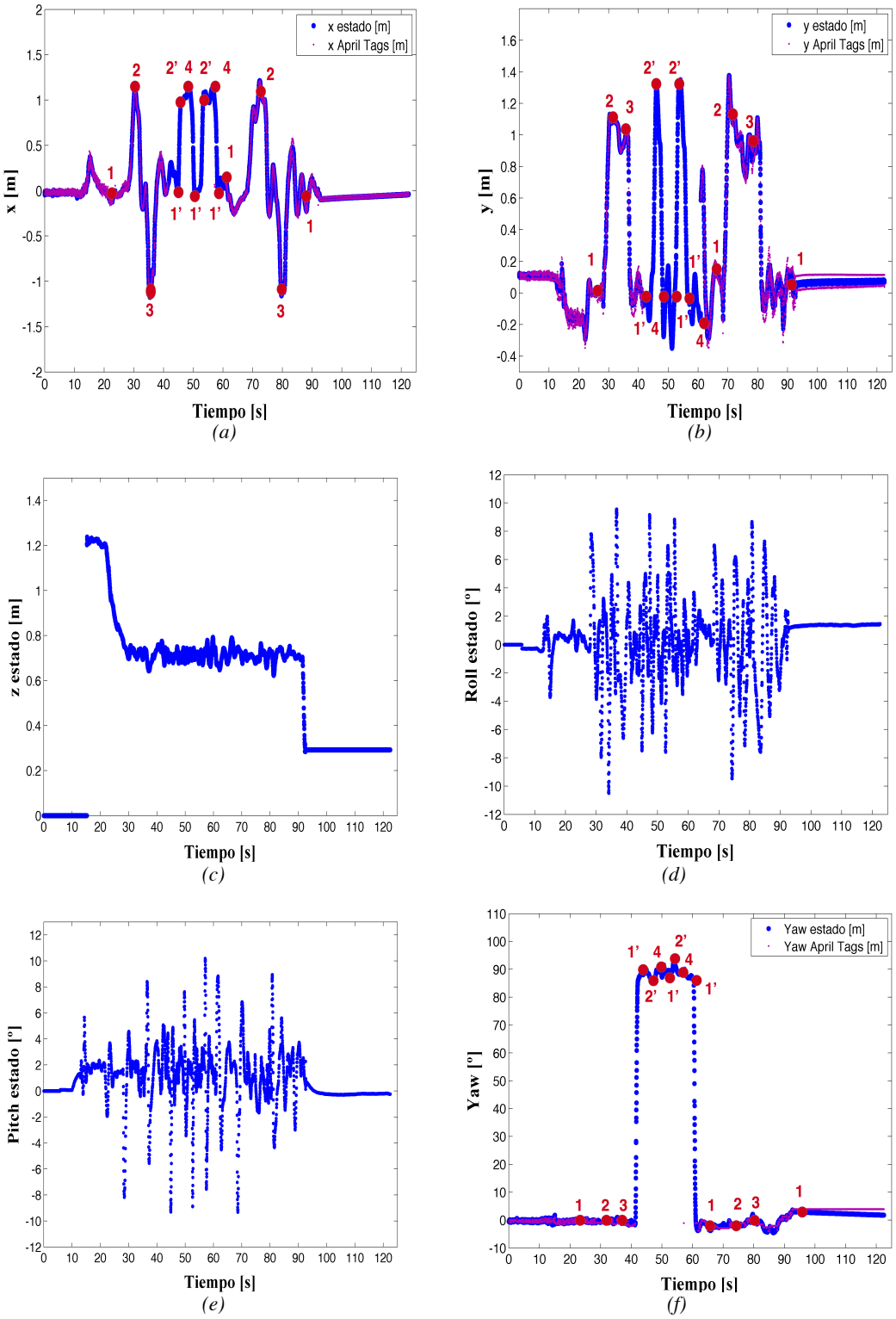
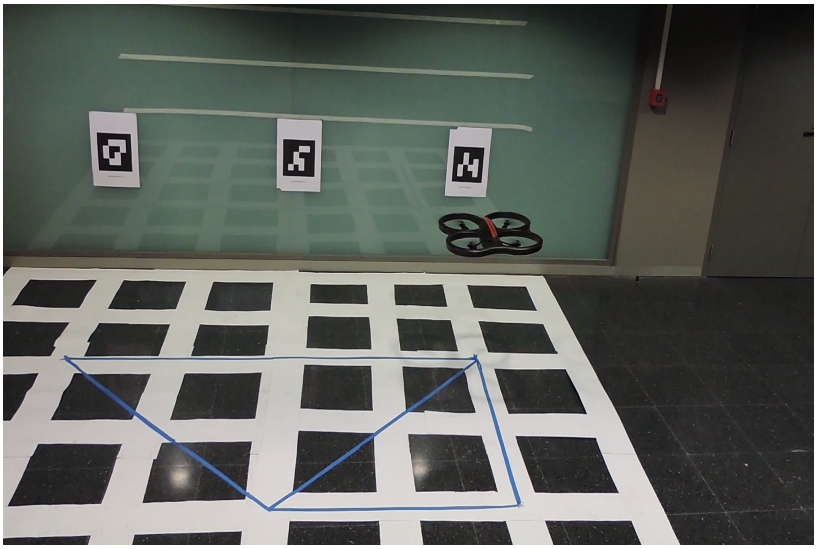
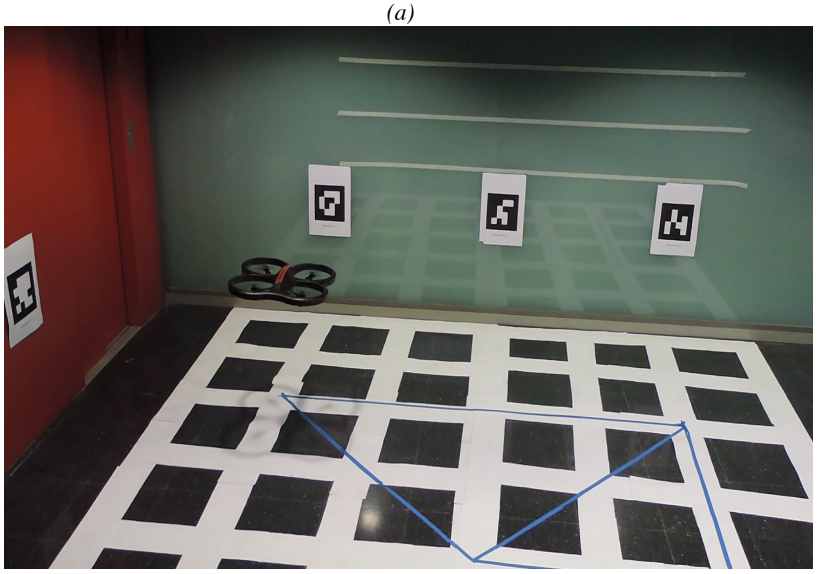


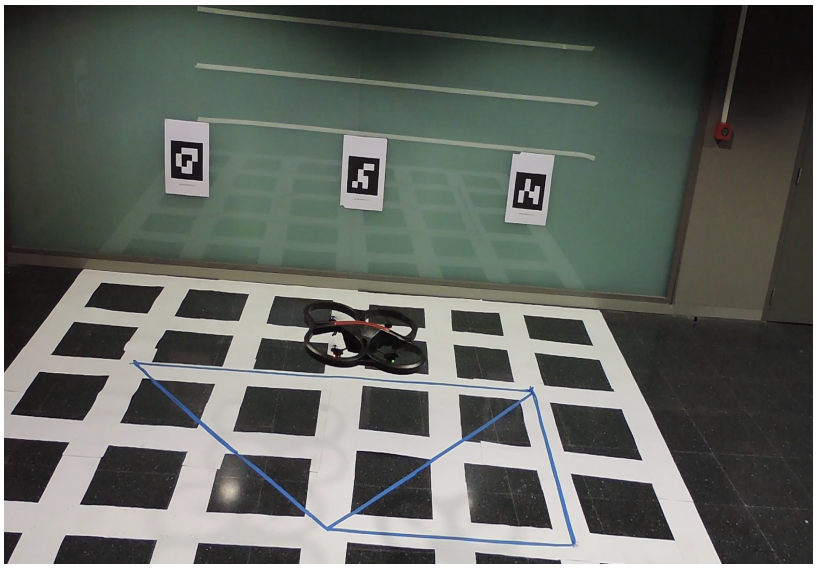
Figura 48. *Representación de las variables de estado del cuadricóptero (en color azul) y de las lecturas del sistema April Tags (en morado) frente al tiempo para la maniobra que se detalla en la tabla 13. (a) Coordenada x. (b) Coordenada y. (c) Coordenada z. (d) Ángulo pitch. (e) Ángulo roll. (f) Ángulo yaw. Las medidas de los April Tags solo se observan en (a), (b) y (f) porque para el resto de variables se han introducido con varianzas muy elevadas que hacen que el filtro no las considere. Se han señalado mediante marcadores circulares de color rojo los instantes en los que el cuadricóptero se encuentra sobre los vértices de la trayectoria (ver figura 47).*



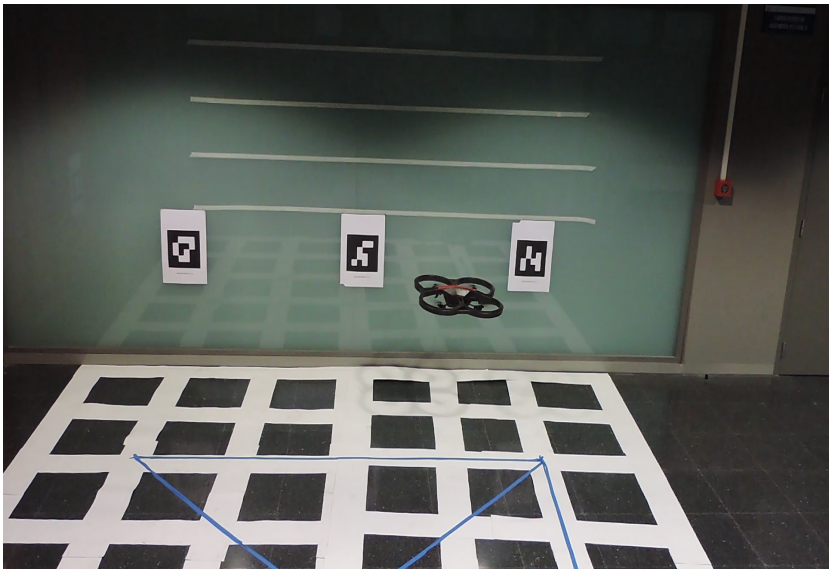
(a)



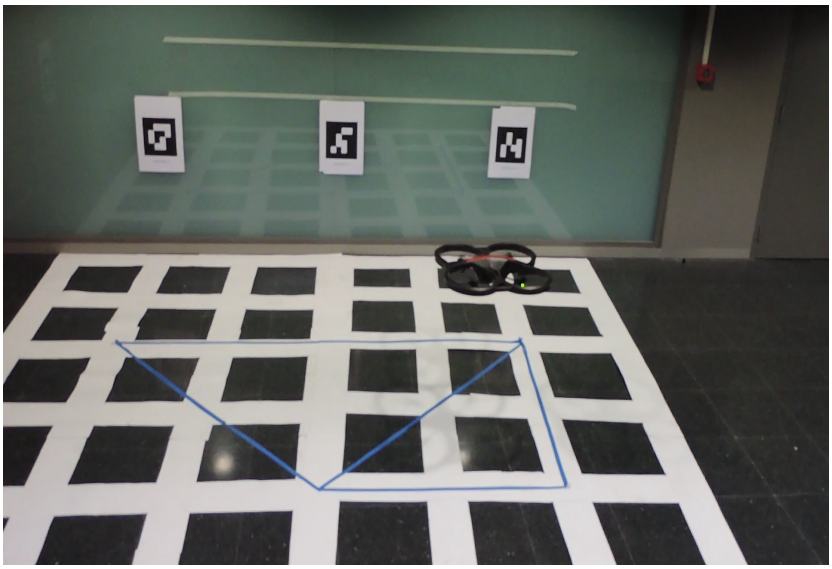
(b)



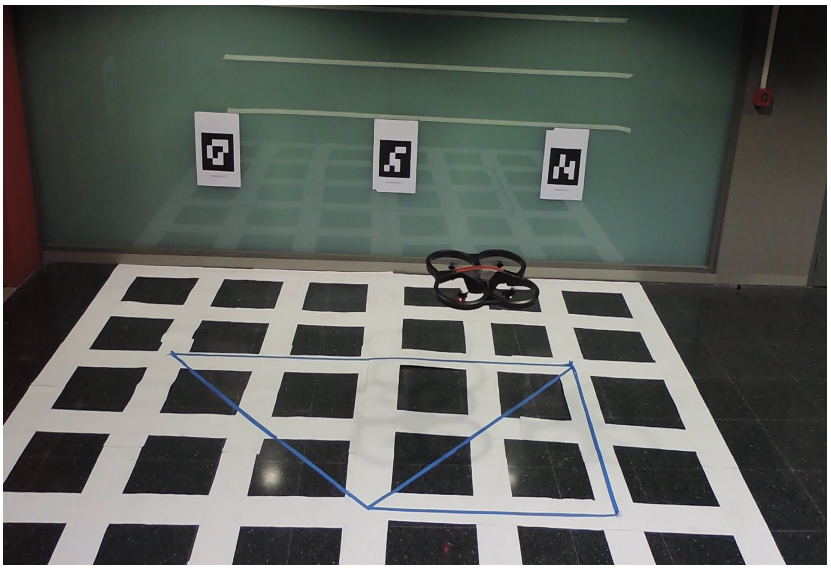
(c)



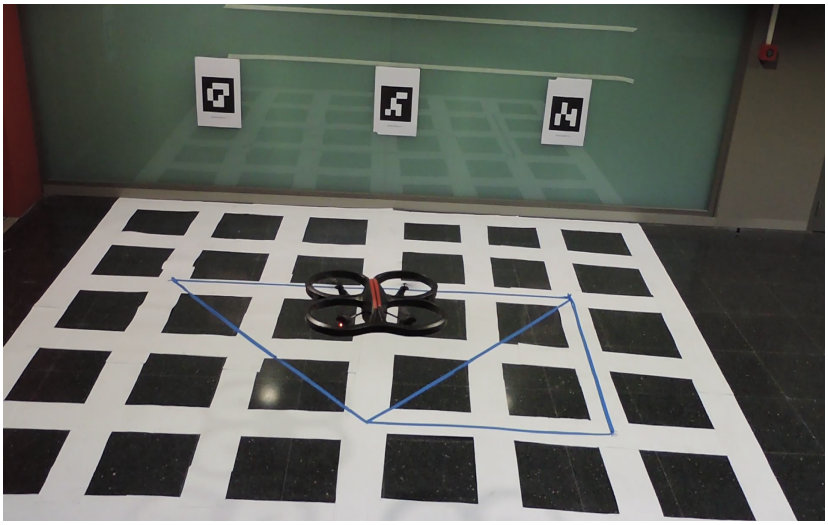
(d)



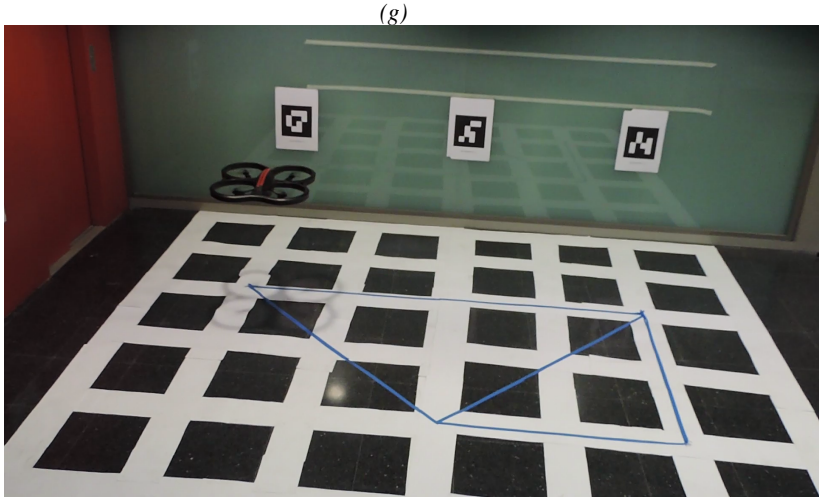
(e)



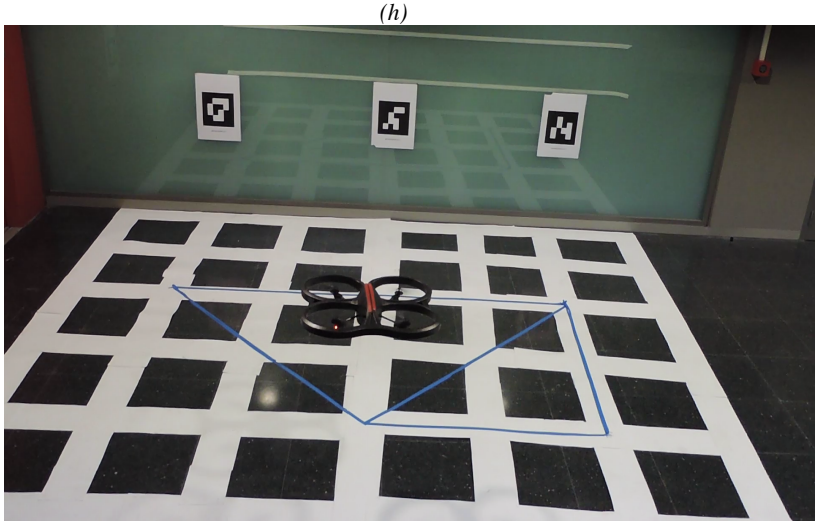
(f)



(g)



(h)



(i)

***Figura 49**. Capturas de video que representan diversos puntos de la maniobra de vuelo. (a) Coordenadas (x,y,z,Ψ) = (1.2,1.2,0.7,0). (b) y (h) Coordenadas (x,y,z,Ψ) = (-1.2,1.2,0.7,0). (c) y (f) Coordenadas (x,y,z,Ψ) = (0,0,0.7,90). (d) Coordenadas (x,y,z,Ψ) = (1.2,1.2,0.7,90). (e) Coordenadas (x,y,z,Ψ) = (1.2,0,0.7,90). (g) e (i) Coordenadas (x,y,z,Ψ) = (0,0,0.7,0).*

Para el punto 3, las lecturas de estado toman un valor de -1.2 m para *x* así como un valor en torno a los 1.1 m para la *y*. Se puede constatar también que, durante toda la maniobra la variable *z* se mantiene oscilando en torno a 0.7 m y que el yaw toma valores muy próximos a cero. Todo ello se corresponde con lo que se observa en la realidad.

- La figuras 49.c y 49.d y 49.e se corresponden con los target *(x,y,z,Ψ) = (0,0,0.7,90), (x,y,z,Ψ) = (1.2,1.2,0.7,90)* y *(x,y,z,Ψ) = (1.2,0,0.7,90)*. Estos conforman los tres vértices (puntos 1', 2' y 4') de la trayectoria triangular que describe dos veces el cuadricóptero sin tener los tags monitorizados. Se observa que para el punto 1 las lecturas de estado oscilan para *x* e *y* en torno a un valor de 0 m. Para el punto 2, las lecturas de estado toman un valor cercano a 1 m para *x* y de aproximadamente 1.3 m para la variable *y*. Por último, para el punto 3, las lecturas de estado llegan a tomar prácticamente el valor de 1.2 m para *x*, y un valor oscilante en torno a 0 m para la *y*. Todo ello diverge bastante de lo que ocurre en realidad, ya que como se observa en las fotografías de la figura 49, el cuadricóptero se desvía bastante de la posición real que debería tener: las coordenadas teóricas *(x,y,z,Ψ) = (1.2,1.2,0.7,90)* se convierten en la realidad en *(x,y,z,Ψ) = (0.8,2.5,0.7,90)* y las coordenadas *(x,y,z,Ψ) = (0,0,0.7,90),* en (x,y,z,Ψ) = (0.3,0.6,0.7,90). También se puede constatar que durante toda esta fase del vuelo, la *z* toma valores muy próximos a 0.7 m y el yaw oscila en torno a los 90º, lo cual sí que concuerda con la realidad.

- La figuras 49.g y 49.h y 49.i se corresponden con los target *(x,y,z,Ψ) = (0,0,0.7,0)*, *(x,y,z,Ψ) = (-1.2,1.2,0.7,0)* y *(x,y,z,Ψ) = (0,0,0.7,0)* de nuevo. Como ya se ha visto, estas coordenadas conforman los tres vértices (puntos 1, 2 y 3) de la segunda trayectoria triangular que describe es el cuadricóptero cuando tiene los tags monitorizados. El punto clave para ver cómo de bien responden los filtros implementados en el apartado 4.5 se encuentra en el punto 1. Como se comentó anteriormente, el estado y la realidad ya no se corresponden después de una serie de maniobras sin ver tags. La posición en *x* e *y* que el estado considera como (0,0) en realidad es de (0.3,0.6). Esta diferencia queda de manifiesto cuando se vuelven a monitorizar los tags: en *x* apenas se nota el salto porque los valores del estado no difieren excesivamente de la realidad. Sin embargo, en la *y* se observa claramente un salto desde a -0.2 m a 0.6 m. Esto pone de manifiesto que el reseteo del estado funciona correctamente y ayuda a que el cuadricóptero se reubique en el verdadero origen. Posteriormente se repite la maniobra triangular que se realizó al comienzo, con una precisión muy similar. Para el punto 3, las lecturas de estado toman un valor de -1.2 m para *x* y de aproximadamente 1.1 m para la variable *y*. Para el punto 1 que marca el final de la maniobra, las lecturas de estado para *x* e *y* toman un valor muy próximo a 0 m. También se puede constatar que durante toda la maniobra la *z* toma valores

muy próximos a 0.7 m y el yaw oscila en torno a los 0º, lo cual se ajusta bastante a la realidad.

- En cuanto al roll y el pitch (figuras 49.d y 49.e respectivamente), se observa que oscila entre valores angulares entre los -10 y 10º. Esto podría llamar la atención, pero se explica porque el dron realiza desplazamientos bastante bruscos en *x* e *y* lo cual implica una cierta inclinación del dron en su desplazamiento.

6. Conclusiones y trabajo futuro.

Tal y como se estableció en la sección de objetivos, en este trabajo fin de grado se ha planteado lograr que el cuadricóptero comercial AR Drone navegue de forma autónoma, es decir, que empleando la información procedente de sus sensores de abordo sea capaz de localizarse a sí mismo en el espacio y de realizar unas maniobras preestablecidas sin que sea necesaria ninguna intervención externa. Dicho objetivo se ha cumplido satisfactoriamente y en este capítulo se presentarán de manera resumida las principales conclusiones y aportaciones de este trabajo.

La primera tarea llevada a cabo ha consistido en el estudio en profundidad de un sistema propuesto por el Grupo de Visión Artificial de la Universidad Técnica de Múnich especialmente desarrollado para el AR Drone, conocido con el nombre de TUM AR Drone y distribuido libremente bajo licencia GNU para su uso en investigación. Dicho sistema proporciona una herramienta básica con la que estimar la posición del dron a partir de la información enviada por la cámara (empleando unos algoritmos de monocular SLAM conocidos como PTAM) y los sensores onboard (IMU, altímetro, etc) del mismo mediante un filtro de Kalman extendido. Además, incorpora también un controlador PID con el que enviar órdenes al cuadricóptero para que se posicione en un determinado punto en base a la estimación de estado proporcionada por el filtro. Como resultado de esta tarea se extrajo una descripción detallada del funcionamiento interno del citado paquete de software, lo que nos dotó de las herramientas necesarias para poder realizar aportaciones y modificaciones a dicho sistema. El trabajo en esta parte se centró en determinar la implementación particular del filtro de Kalman que se estaba empleando, lo cuál se ha documentado en el apartado 4.1. Fruto de esta primera tarea se detectaron una serie de limitaciones asociadas al sistema original que impedían la realización de una navegación autónoma general, relacionadas fundamentalmente con los problemas que presenta el monocular SLAM para reconocer movimientos con una componente de rotación significativa, así como el requerimiento de cierta información adicional de otros sensores para obtener una referencia de escala que no siempre está disponible.

Las limitaciones que se detectaron en el sistema de monocular SLAM llevaron a que se decidiese prescindir de su uso y llevar a cabo una navegación empleando únicamente los sensores onboard del cuadricóptero y la información de los comandos de control enviados por el controlador PID. Se realizaron numerosas pruebas de vuelo empleando únicamente esta información pero se revelaron numerosos problemas que hacían imposible la navegación (deriva, información falsa proveniente de algunos sensores durante el despegue, etc.). Se decidió, por tanto, plantear una optimización del procesado de los datos enviados por los sensores onboard con el objetivo de poder realizar una navegación básica empleando exclusivamente esta información. Dicho

procesado se presenta en la sección 4.2 y condujo a mejoras significativas en el funcionamiento del sistema:

- En primer lugar, se detectó (sección 4.2.1) tras la realización de múltiples experimentos que para la estimación de las velocidades horizontales la textura y elementos característicos del suelo son factores clave que afectan al funcionamiento de los algoritmos de flujo óptico que emplea el AR Drone (junto con información de los acelerómetros de la IMU) para estimar la velocidad. Se llegó a la conclusión de que lo que se ha denominado como suelos tipo I, es decir, bien texturizados y no homogéneos, son los que conducen a una mejor estimación de las velocidades y por tanto mayor precisión en las maniobras. Los suelos tipo 0 (homogéneos y mal texturizados) hacen que resulte prácticamente imposible navegar.

- En segundo lugar, se descubrió (sección 4.2.2) que existe un transitorio en los datos enviados por el altímetro durante el despegue que provocaba que la estimación de altura que resultaba de la aplicación del sistema original planteado en el TUM AR Drone fallase. Su corrección pasó por la implementación de un filtro de tiempos que permitiese eliminar esos valores. También se diseñó un modelo de calibración de las lecturas y se mejoró el criterio de cálculo de alturas relativas con el que eliminar las discontinuidades en la estimación de la altura derivadas de cambios en la cota del suelo.

- En tercer lugar, se estudió (sección 4.2.3) la precisión de las medidas del roll y el pitch que se obtienen de procesar los datos procedentes de los giroscopios *x* e *y* de la IMU. Se concluyó que la precisión era susceptible de mejora, la cual se logró mediante la implementación de un modelo de calibración.

- En cuarto lugar, se estudió (sección 4.2.4) la precisión de las medidas del yaw que se obtienen de procesar los datos procedentes del giroscopio en *z* de la IMU. Se observó que las lecturas antes del despegue están sujetas a una significativa deriva. Para eliminarla se implementó un filtro de tiempos y se mejoró el cálculo del yaw de forma relativa que se planteaba en el sistema original.

La realización de este proceso de optimización permitió que se mejorase enormemente la capacidad de navegación del cuadricóptero de manera fiable usando exclusivamente sus sensores onboard durante un tiempo mucho mayor. Sin embargo, y con el objetivo de mejorar la calidad de la navegación, se ha tratado de eliminar la deriva que aparece en *x* e y debido a los errores que se acumulan durante el proceso de odometría, así como la deriva intrínseca que posee el giroscopio en *z*.

Para lograrlo, se ha optado por incluir un sistema en base a marcadores artificiales conocido como April Tags Fiducial System, que proporciona la posición de un determinado tag con respecto al centro la cámara que lo observa. Una parte de este trabajo (sección 4.3) se dedicó a la transformación de las lecturas que devuelve este sistema en un "sensor" fiable del estado del cuadricóptero. El procesado implementado para lograrlo se puede dividir en dos operaciones fundamentales: la primera de ellas (apartado 4.3.1) se corresponde con la realización de las transformaciones algebraicas que nos permitan convertir las lecturas del April Tags Fiducial System proporcionadas en el sistema cámara a una lectura que represente la posición y orientación del cuadricóptero en el sistema de referencia mundo. La segunda consiste en la realización de un estudio de las medidas que se obtienen del April Tags Fiducial System a nivel de precisión y exactitud (apartado 4.3.2). Dicho estudio se divide en tres tareas. En primer lugar se plantearon una serie de experimentos (apartado 4.3.2.1) en los que se colocó el cuadricóptero en diferentes posiciones y se guardó un log de mediciones para su posterior análisis. En segundo lugar se llevó a cabo un proceso de filtrado de las medidas (apartado 4.3.2.2). El filtro creado ha demostrado ser altamente efectivo, ya que aumenta en gran medida la exactitud de las medidas, llegando a reducirse la desviación estándar incluso en un orden de magnitud. Por último, se observa que a pesar del filtrado persiste un error sistemático que indica la necesidad de un proceso de calibración (apartado 4.3.2.3). Se escogió un modelo lineal multivariable para ajustar cada una de las lecturas y el resultado de su aplicación fue bastante bueno, ya que se redujo el error cometido en más de la mitad de las medidas para cada variable así como el error máximo para cada una de ellas.

También se hizo necesario el integrar las medidas procedentes del procesado de las lecturas de los sensores onboard con las que provienen del procesado de las lecturas de los April Tags. Esto se presenta en la sección 4.4 e implicó la sincronización de los datos (ya que cada una de las citadas fuentes de información los envía con distinta frecuencia) y el diseño de unos filtros *online* en los que se combinan las dos fuentes de información para mejorar la precisión de las medidas.

Finalmente, se realizaron una serie de pruebas de validación (sección 5) para comprobar el correcto funcionamiento del sistema proyectado, evaluándose diferentes aspectos de la navegación: vuelo en presencia de obstáculos, realización de maniobras que entrañan numerosos cambios de orientación y vuelo donde se alternan periodos en los que la monitorización de los tags esta disponible con otros en los que no. Se constató la fiabilidad y robustez del sistema de navegación diseñado, con un nivel de error en la estimación de la posición muy satisfactorio que supondrá una herramienta para futuros trabajos a los investigadores del Grupo Integrado de Ingeniería de la UDC, que permitirá seguir avanzando en la línea de robótica autónoma.

Como posibles líneas de mejora a los resultados alcanzados en este trabajo fin de grado se proponen las siguientes:

- Mejora del modelo de calibración: el modelo lineal multivariable escogido es el que más reduce el error de todos cuantos se han probado. Sin embargo su funcionamiento es mejorable ya que a pesar de que en general mejora la estimación, hay un gran número de medidas en las que el error empeora al aplicar la calibración.

- Desarrollo de un controlador de alto nivel inteligente: este trabajo ha sentado las bases del sistema de navegación centrándose en el aspecto de conseguir una buena estimación de estado. El sistema de control es una parte que todavía queda por explorar en profundidad y en la que se ha puesto la primera piedra, que es el haber creado un controlador básico que permite la comunicación con el PID mediante órdenes de alto nivel.

- Navegación colectiva: se pretende utilizar este mismo sistema de navegación simultáneamente en una flota de drones, lo cual ampliará notoriamente las posibilidades a la hora de llevar a cabo tareas complejas y mejorará las capacidades de vuelo, permitiendo estudiar la posibilidad de realizar navegación colectiva incluyendo tags en la carcasa de los drones.

- Implementar el uso de balizas naturales: continuando con la idea original detrás del uso de las librerías PTAM, se plantea seguir estudiando diversas alternativas que puedan permitir incorporar el uso de balizas naturales que no estén supeditadas a las limitaciones del PTAM, lo cuál conduciría a minimizar la intervención de los diseñadores en la navegación y posibilitará el uso de los drones en entornos desconocidos o inaccesibles.

Anexo I: ROS

El objetivo de este anexo es dar una idea general de como funciona ROS, entrándose a explicar más en detalle cada uno de los elementos que lo conforman. Se consideran tres niveles dentro de esta plataforma: el de sistema de archivos, presentado en el apartado 1, el de computación, presentado en el apartado 2, y el de comunidad, presentado en el apartado 3.

1. Nivel de sistema de archivos

Este nivel incluye aquellos conceptos que se encuentran a nivel de disco, tales como:

Paquetes

El software de ROS está organizado en paquetes. Un paquete puede contener un nodo de ROS, una librería independiente, un conjunto de datos, archivos de configuración, software de terceros o algo que por sí mismo puede constituir un módulo. El objetivo de estos paquetes es que el software sea lo más útil posible pero sin caer en una complejidad tal que haga difícil su adaptación a otros tipos de software.

Los paquetes son muy sencillos de crear a mano, aunque se suele emplear una herramienta llamada *catkin_create_pkg*. A nivel técnico, un paquete es simplemente un directorio con un fichero *package.xml* en su interior que desciende de ROS_PACKAGE_PATH, que es una variable de entorno cuya función es indicar a ROS como localizar los paquetes y stacks dentro del sistema de archivos. Los paquetes suponen la unidad atómica de ROS, es decir, que es la unidad individual más pequeña que puedes compilar y es la forma en la que el software se empaqueta cuando es publicado.

Metapaquetes

Un metapaquete es un tipo especializado de paquete en ROS, que no instalan archivos (aparte de el manifiesto *package.xml* que contienen) y no contienen pruebas, código ni otros elementos que se encuentran en los paquetes. Básicamente sirven para referenciar uno o más paquetes que están relacionados entre sí. La mayoría de los metapaquetes en ROS lo que hacen es funcionar como representantes de los stacks compilados.

Manifiestos de paquetes

Un manifiesto de un paquete es un archivo XML llamado *package.xml* que debe ser incluido en la carpeta raíz de cualquier paquete. Este archivo define las propiedades del paquete, tales como en nombre del mismo, su versión, los autores, personas encargadas de su mantenimiento y las dependencias de otros paquetes. Si por ejemplo las dependencias de un paquete en cuestión con respecto a otros faltan o son incorrectas, es

posible que podamos compilar el código y llevar a cabo pruebas en nuestro propio ordenador, pero el paquete no funcionará correctamente cuando lo publiquemos en la comunidad ROS.

Repositorios

Se trata de una colección de paquetes que comparte un mismo VCS (Version Control System). Esto significa que los paquetes comparten la misma versión y pueden ser publicados juntos usando la herramienta automática conocida como *bloom,* que básicamente transforma nuestro código fuente en un repositorio git.

Tipo de mensajes

Se trata de una descripción simplificada de las estructuras de datos de los mensajes publicados por los nodos de ROS. Esto hace que sea mucho más sencillo para las distintas herramientas de ROS el generar código fuente para el tipo de mensaje en diferentes lenguajes de programación. Estas descripciones se almacenan en archivos *.msg* en el subdirectorio *msg/* de un paquete de ROS.

Estos archivos constan de dos partes: los campos y las constantes. El primer concepto se refiere a la información que se envía dentro del mensaje. El segundo define valores útiles que pueden ser utilizados para interpretar dicha información (por ejemplo constantes *enum* para un valor entero).

Tipos de servicios

Se trata de un concepto similar al anterior, pero en este caso suponen una descripción simplificada de las estructuras de datos de los servicios de ROS. Esto permite simplificar los mecanismos de petición/respuesta entre nodos de ROS. Estas descripciones se almacenan en archivos *.srv* en el subdirectorio *srv/* de un paquete de ROS.

2. Nivel de computación

A este nivel se le denomina *Grafo de Computación*, y es una red peer-to-peer (P2P) de los procesos de ROS que están trabajando juntos sobre una serie de datos. El concepto P2P implica que intercambian recursos entre ellos sin necesidad de un sistema administrativo centralizado que gestione este intercambio. Dentro de este nivel tenemos que describir una serie de conceptos clave:

Nodos

Un nodo es un proceso que lleva a cabo tareas de computación. Los nodos se combinan entre ellos para forman un grafo, y se comunican entre ellos usando topics, servicios y

un servidor de parámetros. Los nodos están diseñados para funcionar de forma precisa, estando los sistemas robóticos conformados por varios nodos por lo general: un nodo que procesa datos de un determinado sensor, otro que se encarga de la localización, otro de la planificación, etc.

Esta división de los procesos en nodos es una gran ventaja que ofrece ROS, puesto que si el sistema falla, el error va a ser más fácil de localizar y aislar, posibilitando el llegar antes a la solución. También se reduce la complejidad del código con respecto a sistemas monolíticos, ya que se puede dividir el código en partes más pequeñas que realizan tareas concretas que luego ROS nos permite interrelacionar. Esto a su vez facilita la implementación y la reutilización del código, debido a que si en nuevos desarrollos no necesitamos o queremos cambiar la función de algún nodo concreto, va a resultar mucho más sencillo que si el sistema fuese monolítico.

Todos los nodos que están en ejecución tienen un *nombre recurso del grafo* que los identifica dentro resto del sistema. Los nodos también tienen un tipo que simplifica el proceso de referirse a un ejecutable del nodo dentro del sistema de archivos. Estos tipos se representan como *nombres recursos del paquete*, con el nombre del paquete del nodo y el nombre del archivo ejecutable del nodo. Para resolver el tipo de un nodo, ROS busca todos los ejecutables del paquete con el nombre especificado y escoge el primero que encuentre. Es por eso importante no tener diferentes ejecutables con el mismo nombre en un mismo paquete.

Maestro

El Maestro de ROS proporciona los servicios de nombramiento y registro al resto de nodos del sistema. Se encarga de buscar los nodos que publican y se subscriben a los topics, así como los servicios activos. Su papel es pues que los distintos nodos puedan localizarse entre si. Una vez hayan logrado localizarse, van a poder comunicarse a través de una red peer-to-peer.

Servidor de parámetros

Se trata de un diccionario compartido y multivariable que es accesible a través de redes API. Los nodos utilizan este servidor para almacenar y recuperar parámetros en el tiempo de ejecución. No está diseñado para dar prestaciones de alto rendimiento, por lo que es más adecuado para datos estáticos y no binarios, tales como parámetros de configuración. Está pensado para ser visible de forma global, de forma que las distintas herramientas puedan inspeccionar el estado de la configuración del sistema y modificarlo si es necesario.

Mensajes

Los nodos se comunican entre ellos mediante la publicación de mensajes en los topics. Un mensaje es simplemente una estructura de datos que contiene un determinado campo de tipos. Los tipos primitivos tales como enteros, punto flotante, booleanos, etc. están soportados, así como arrays de estos tipos y estructuras anidadas.

Topics

Se trata básicamente de buses de datos a los que se les asigna un nombre y a través de los cuales los distintos nodos intercambian mensajes. Los topics presentan una semántica de publicación/subscripción anónima, lo que permite desacoplar la producción de información de su consumo. Esto implica que por lo general lo nodos no saben con quién se están comunicando. En lugar de eso, los nodos que están interesados en adquirir un determinado tipo de información relevante para el proceso que llevan a cabo (es decir, que la consumen), han de suscribirse al topic pertinente. Del mismo modo, los nodos que generan un determinado tipo de información publicarán en el topic pertinente.

Es importante comentar que pueden existir múltiples publicadores y subscriptores para un mismo topic, y que estos están pensados para una comunicación unidireccional en streaming. Los nodos que necesitan por ejemplo, recibir una respuesta a una determinada petición, deberían usar servicios en lugar del topic.

Servicios

Como se acaba de comentar, el modelo de publicación/subscripción a través de los topics es muy flexible pero no es adecuado para interacciones petición/respuesta, que son de uso común en los sistemas distribuidos. Para este tipo de funciones es para lo que existen los servicios, que están definidos por un par de mensajes: uno de petición y otro de respuesta. ROS proporciona un nodo que ofrece el servicio bajo un nombre tipo string, y el cliente llamará al servicio mediante el envío de un mensaje de petición y esperando por la respuesta.

Bags

Se trata de un formato de archivos en ROS destinado al almacenaje de los datos contenidos en los mensajes. Se trata de un componente muy importante del sistema, ya que el almacenaje de datos nos permite un posterior procesado, análisis, y visualización de los mismos mediante las distintas herramientas que ROS pone a nuestra disposición (por ejemplo, con la herramienta rxplot podemos graficar los datos que ha publicado el nodo encargado de gestionar los datos de un sensor láser).

Con el objetivo de enlazar todos estos conceptos se va a explicar de manera resumida como funciona ROS a este nivel: en primer lugar tenemos el Maestro, que almacena información sobre los registros de servicios y topics para los nodos de ROS. Estos últimos se comunican con el Maestro para pasarle su información de registro. Del mismo modo que envían, también pueden recibir información sobre otros nodos registrados y hacer las conexiones adecuadas según demande la situación. El Maestro también puede hacer callbacks (retrollamada) a estos nodos cuando la información de registro cambia, lo que permite a los nodos existentes crear dinámicamente nuevas conexiones con los nuevos que van apareciendo en el tiempo de ejecución.

Es importante destacar que los nodos se conectan con otros nodos directamente, siendo el Master únicamente un guía por así decir, que proporciona exclusivamente información de búsqueda (el principio de funcionamiento es análogo al de un servidor DNS). Los nodos que se suscriben a un topic solicitarán conexiones con los nodos que publican en dicho topic, y dicha conexión se realizará según un determinado protocolo. El más común es el TCPROS, que utilizada zócalos TCP/IP. Esta arquitectura permite desacoplar la operación, siendo los nombres los principales medios de referencia por los que se pueden llevar a cabo sistemas más complejos. Todas las librerías de cliente de ROS permiten emplear un remapeado de nombres por línea de comandos, lo que significa que un programa ya compilado puede ser reconfigurado durante el tiempo de ejecución para operar con otra topología del grafo de computación.

Por poner un ejemplo aplicado a un cuadricóptero, supongamos que tenemos el sensor para medir la altura (altímetro) del AR Drone. Podemos iniciar un *ardrone_node* donde se ejecute un driver, que lo que hará sería comunicarse con el altímetro y publicar la información que reciba de este en sensor como mensajes en *sensor_msgs/altimeter* en el topic *altimeter*. Para procesar esos datos necesitaremos escribir un nodo (*altimeter_filter* por ejemplo) que lo que haga sea suscribirse a los mensajes que se publiquen en el *altimeter* topic. Se puede observar como los procesos están totalmente desacoplados: por un lado el *ardrone_node* publica mensajes con la información proporcionada por el altímetro en *altimeter,* sin tener conocimiento alguno de si hay alguien suscrito a ese topic. Por otro lado, el *altimeter_filter* se suscribe al topic sin tener conocimiento alguno de si alguien esta publicando en el mismo. Los dos nodos pueden iniciarse, cerrarse y reiniciarse en cualquier orden sin conducir a error en el proceso. Si por ejemplo posteriormente queremos añadir otro sensor de altitud a nuestro robot, vamos a tener que reconfigurar este sistema. Para ello, todo lo que tenemos que hacer es remapear los nombres que están siendo usados. Cuando iniciemos de nuevo el *ardrone_node* podemos indicarle que remapee el topic altimeter a altimeter_base y que altimeter_filter pase a ser altimeter_filter_base, y usar los nombres sin base para el nuevo sensor. De este modo hacemos tenemos dos topics independientes, pudiendo repetirse el proceso para añadir cuantos nuevos sensores queramos.

3. Nivel de comunidad

En este nivel se encuentran los conceptos relativos a los recursos que permiten a distintas comunidades de usuarios compartir software y conocimientos a través de la Red. Los principales son:

Distribuciones

Las distribuciones de ROS son colecciones de stacks (conjunto de paquetes de ROS, es decir, es una unida básica de software a la hora de compartirlo) con una determinada versión asociada y con sus dependencias con otros stacks. Básicamente el objetivo de estas distribuciones es idéntico al de las de Linux: facilitar el proceso de instalación de una determinada colección de software y el mantener una consistencia en las versiones de cierto conjunto de software.

Repositorios

ROS tiene una red federada de repositorios de código, en la que diferentes instituciones pueden desarrollar y publicar su propio software para su aplicación a un determinado tipo de robot. Esto implica que es la propia persona o institución la que se encarga de gestionar y establecer las licencias, reteniendo en todo momento la propiedad y control sobre el código.

La Wiki de ROS

La comunidad de usuarios de ROS mantiene una página web del tipo Wiki, constituyendo esta el principal foro para documentar cualquier aspecto de ROS. Cualquier persona puede registrarse y crear su propia cuenta, y es libre de contribuir con su propia documentación, aportar correcciones o actualizaciones del software existente, escribir tutoriales, etc.

Listas de Email

Existe una lista de los correos electrónicos de los usuarios de ROS, siendo este el principal medio de comunicación sobre las novedades en la plataforma.

Respuestas de ROS

Se trata de un foro del tipo pregunta/respuesta en el que se tratan de dilucidar todas las cuestiones relacionadas con ROS que le puedan surgir a sus usuarios.

Blog

Es un medio en el que se presentan regularmente las novedades en la plataforma ROS, con fotos, vídeos, comentarios, etc.

Anexo II: TUM AR Drone

El objetivo de este anexo es explicar de forma detallada como funcionan los distintos elementos que componen el paquete de software TUM AR Dron: el sistema de monocular SLAM, que se presenta en el apartado 1, el filtro de Kalman extendido, presentado en el apartado 2, y el controlador PID, que se presenta en el apartado 4.

1. Monocular SLAM

Para abordar el problema del SLAM, es necesario obtener información de los sensores, los cuales pueden ser clasificados en dos grupos: visuales, como por ejemplo una cámara monocular o una estéreo cámara, y no visuales, como sensores basados en láser o ultrasonidos. La formulación matemática del problema del SLAM es independiente del tipo de sensor que empleemos, pero está claro que se simplificará más cuanto mayor sea la información. Por ejemplo, las estéreo cámaras son capaces de medir la profundidad, lo cual no es posible con una cámara monocular. Sin embargo, esta última es la que se usa más comúnmente puesto que su precio es notoriamente inferior al de la primera.

En los sistemas basados en SLAM, el mapa del entorno se crea en base a un determinado número de marcadores (*landmarks*), que pueden ser por ejemplo puntos en el espacio tridimensional reconocibles y localizables en la imagen obtenida mediante la cámara. En base a la posición de estos puntos en la imagen, la posición de la cámara puede ser estimada.

A la hora de resolver el problema del SLAM existen dos enfoques fundamentales: uno es el enfoque basado en el filtrado, como pueden ser el EKF-SLAM o el Fast-SLAM. En ellos, lo que se hace es aplicar una distribución de probabilidad sobre la posición de los marcadores ($x_1, x_2, ..., x_n \in \mathbb{R}^3$) y de la cámara ($C \in SE(3)$) generando una distribución gaussiana multivariable con media $x = (x_1, x_2, ..., x_n, C) \in \mathbb{R}^{3n+6}$ y covarianza $\sum \in \mathbb{R}^{(3n+6)x(3n+6)}$. Los marcadores aparecen como regiones (*patches*) de la imagen. El otro enfoque, que va a ser el de interés en este trabajo es el que se basa en keyframes. El sistema más conocido basado en este enfoque es el PTAM**,** desarrollado por G.Klein y D.Murray en 2007 [19].

1.1 Paralell Mapping and Tracking (PTAM)

El PTAM parte la posición de los marcadores ($x_1, x_2, ..., x_n \in \mathbb{R}^3$), la posición de los keyframes (capturas de imagen tomadas cada cierto tiempo que sirven de referencia debido al elevado número de marcadores que aparecen en ellas, llevando asociado un

sistema de coordenadas centrado en la cámara denotado como $K_1,...,K_m \in SE(3)$), las respectivas imágenes de la cámara ($I_1,...,I_m$), y por último de las observaciones de los marcadores en los keyframes, que viene a ser lo que se conoce como keypoint ($\tilde{p}_{ij} \in \mathbb{R}^2$) con $i = 1,...,n$ y $j = 1,...,m$.

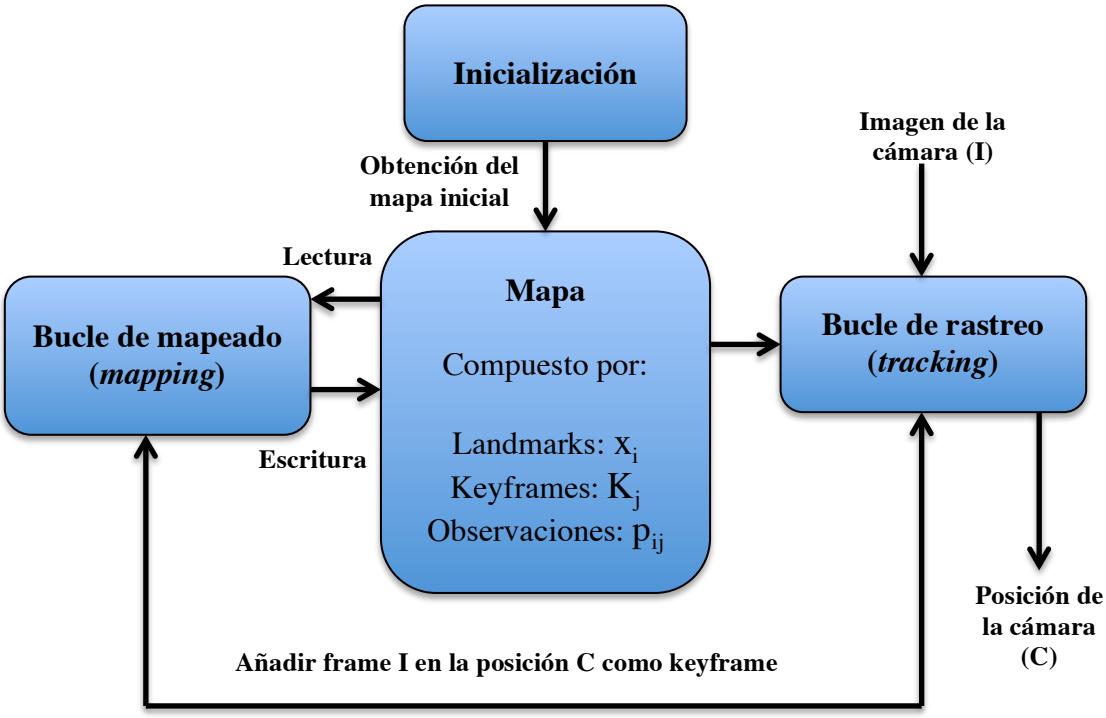


***Figura 50**. Esquema de funcionamiento del PTAM.*

El algoritmo del PTAM consta de tres fases bien diferenciadas que se resumen en la figura 50. A continuación pasan a detallarse cada una de ellas:

1.1.1 Inicialización

Requiere de algún tipo de movimiento de la cámara, como puede ser una translación en paralelo al plano de la imagen. Este paso tiene como entrada los frames iniciales de video y da como salida el mapa inicial. Una de las principales dificultades del enfoque basado en keyframes es se que genera un problema del tipo "pescadilla que se muerde la cola", ya que para construir un mapa se necesita poder seguir el movimiento de la cámara, lo que implica a su vez tener un mapa previo en el que basarnos, e inicialmente no tenemos ninguno. Este problema no existe, por ejemplo, si tenemos una estereocámara, ya que tenemos información sobre la profundidad y puede construirse un mapa directamente de la primera imagen. Para cámaras monoculares el proceso de inicialización consiste en lo siguiente:

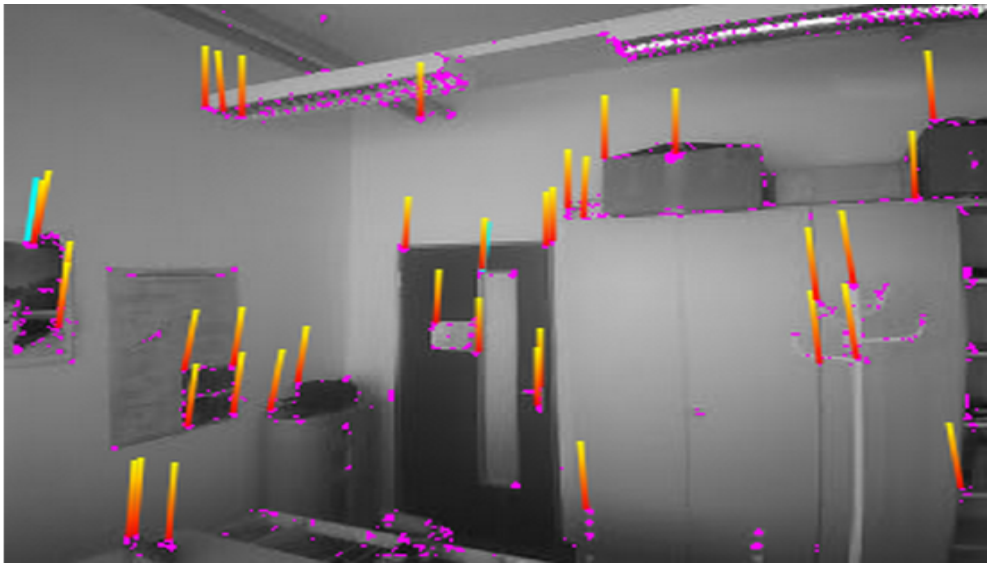


***Figura 51**. Representación de cómo se ve el proceso de inicialización del PTAM en la imagen. Cada línea se corresponde con un keypoint captado con éxito, mostrando cuál es su recorrido desde su localización en el primer keyframe hasta su localización en el keyframe actual [21].*

En primer lugar se e toma el primer keyframe K_1 y se detectan keypoints ($p_1,...,p_n$) empleando el detector de esquinas tipo FAST (*FAST Corner Detector*). Este algoritmo fue presentado por Rosten y Drummond en 2006 [20] y se basa en la detección directa de esquinas. Para comprobar si un determinado pixel es el centro de una esquina, lo que hace es observar los pixeles alineados en un círculo a su alrededor. Si resulta que hay una secuencia lo suficientemente grande de pixeles continuamente más brillantes, o continuamente más oscuros, ese punto se considera como una esquina (ver figura 52). Los autores han creado un código de libre acceso que consiste fundamentalmente en un árbol de decisión basado en una gigantesca construcción if-then-else que establece para cada pixel de la imagen si se trata o no de una esquina.

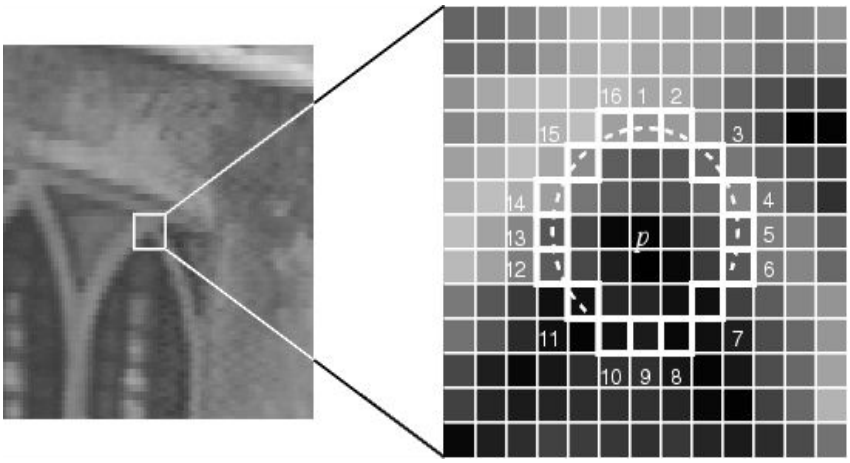


***Figura 52.** Representación visual de cómo funciona el FAST – SLAM [30].*

El segundo paso es el seguimiento de keypoints (*tracking*) hasta que la cámara se haya movido lo suficiente como para que las posiciones de los marcadores puedan triangularizarse. El seguimiento de un keypoint es la tarea de encontrar la posición exacta (y si es posible otros parámetros como la escala y la orientación) de un determinado keypoint en una imagen. Matemáticamente, el proceso de tracking se formula como la búsqueda de un parámetro p de la función deformación (warp function) definida como $f(x,y;p):\mathbb{R}^2\times\mathbb{R}^d\rightarrow\mathbb{R}^2$ tal que la diferencia entre una determinada región (*patch*) de la imagen $T(x,y)$ y la imagen transformada $I(f(x,y,p))$ se haga mínima:

$$p^* = \arg\min E_{SSD}(p) \qquad [73]$$

Donde $E_{SSD}(p)$es la suma de las diferencias cuadráticas:

$$E_{SSD}(p) = \sum_{x,y}\left(I(f(x,y,p)) - T(x,y)\right)^2 \qquad [74]$$

La función deformación puede tomar diversas formas, pero en el PTAM se utiliza una función que viene a ser una translación pura con respecto al keyframe donde el keypoint fue detectado por primera vez. Esto es suficiente cuando se hace un seguimiento frame a frame como es el caso. Esta función toma una forma tal como:

$$f(x,y,\delta\mathrm{x},\delta y) = \begin{pmatrix} x+\delta\mathrm{x} \\ \mathrm{y}+\delta y \end{pmatrix} \qquad [75]$$

A continuación se toma un segundo keyframe K_2 y se extraen empleando el FAST – SLAM nuevas posiciones para los keypoints ($p_1^{'},\ldots,p_n^{'}$).

El último paso consiste en generar un mapa inicial a partir de las correspondencias entre los keypoints. La escala del mapa así como la posición y orientación del primer keyframe se hacen de forma arbitraria, tomándose normalmente la unidad como valor para la escala. A la hora de obtener el mapa es fundamental obtener la matriz esencial (essential matrix), que básicamente es una matriz de transformación de un sistema de referencia a otro, permitiéndonos capturar la relación entre dos imágenes de una misma escena pero tomadas bajo diferentes perspectivas. Por ejemplo, si queremos pasar un marcador del sistema mundo a un determinado keyframe K_2 se emplearía una matriz esencial denotada por E_{K_2} . La matriz esencial se define como:

$$E = R\left[t\right]_\times \in \mathbb{R}^{3\times3} \qquad [76]$$

Donde *R* es la matriz de rotación y $[t]_\times$ es la matriz de producto vectorial con *t* que sería el vector traslación. Se cumple que para cada par de observaciones correspondientes $\tilde{p}, \tilde{p}' \in \mathbb{R}^2$ pero tomadas bajo distintas perspectivas que $\tilde{p}'^T E \tilde{p} = 0$ así como otras propiedades, siendo la más importante de ellas que se trata de una matriz que tiene dos valores singulares iguales y un tercero que es igual a cero. Si se interpreta como un elemento proyectivo, E solo puede definirse dependiente de la escala, y tiene 5 grados de libertad y tres restricciones internas que pueden expresarse como $2EE^T E - tr(\mathrm{EE}^T)E = 0$, resultando un total de 9 ecuaciones. Como se ve pues, la matriz tiene seis grados de libertad, siendo 5 posibles de estimar mediante correspondencias entre observaciones, y dejando el sexto (la escala) indeterminado. Por lo general, el método más empleado para calcular E es el algoritmo de los ocho puntos, que está disponible en Open CV [33] y que requiere un mínimo de 8 correspondencias punto a punto. Para cada par $\tilde{p}, \tilde{p}'$ la relación $\tilde{p}'^T E \tilde{p} = 0$ conlleva una restricción lineal entre las entradas de E. Lo que se hace es plantear un sistema tal como:

$$Ae = 0 \qquad [77]$$

Donde $A \in \mathbb{R}^{n\times 9}$ es una matriz que contiene las restricciones y $e \in \mathbb{R}^9$ contiene las entradas de E. En ausencia de ruido y para $n \geq 8$ tendremos una única solución, dependiente de la escala y que se corresponde con el kernel de A. Sin embargo, lo más común es que haya ruido y tengamos $n > 8$, lo que nos lleva siempre a la solución trivial $e = 0$. Para evitar esto y obtener una estimación de e lo que se plantea es el $\min_e Ae$ sujeto a que el módulo de e sea la unidad.

Para resolver el problema de minimización se hace una descomposición en valor singular de A, siendo la solución el vector singular de que corresponde al menor de todos los valores singulares. Una vez estimada E, se pueden sacar de ella tanto R como t. Si consideramos la descomposición en valor singular de E como $E = U\Sigma V^T$ y resolvemos algebraicamente se llega a que existen cuatro posibles soluciones:

$$[t]_\times = \pm VW\Sigma V^T \text{ y } R = UW^{-1}V^T$$

$$\text{Siendo } W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} o\, W = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Se demuestra que solo una de las cuatro soluciones es válida. Una vez conocidos R y t la posición de un determinado marcador x puede ser triangularizada. El valor más probable de x puede ser calculado minimizando el error de reproyección:

$$x^* = \arg\min_x E_{rep}(x, R, t) \qquad [78]$$

$$E_{rep}(x, \mathrm{R}, \mathrm{t}) = \left|\left| proj(K_{cam} x) - \tilde{p} \right|\right|^2 - \left|\left| proj(K'_{cam}(x - t)) - \tilde{p} \right|\right|^2 \qquad [79]$$

Siendo K_{cam} la matriz de proyección de la cámara. Es importante destacar que en presencia de ruido el mínimo de la función error no es un mínimo global, sino local, por lo que no se corresponde con la solución óptima. En base a esta consideración y con el objetivo de tratar de conseguir una mejor estimación, lo que se hace es aplicar una minimización no lineal:

$$\min_{x_1,\ldots,x_n,R,t} \sum_{i=1}^{n} E_{rep}(x_i, R, t) \qquad [80]$$

1.1.2 Mapeado

El bucle del mapeado lo que hace es estar optimizando continuamente el mapa y lo extiende incorporando nuevos keyframes y marcadores. Dadas una serie de observaciones p_{ij} el objetivo es refinar la posición de los keyframes y de los marcadores de modo que estos coincidan lo máximo posible con las observaciones. Considerando que vamos a tener ruido gaussiano independiente en cada observación, la solución más probable se obtiene de nuevo reduciendo el error de reproyección total. Si consideramos el error de reproyección de una única observación p de un marcador x_W desde una posición de cámara C como:

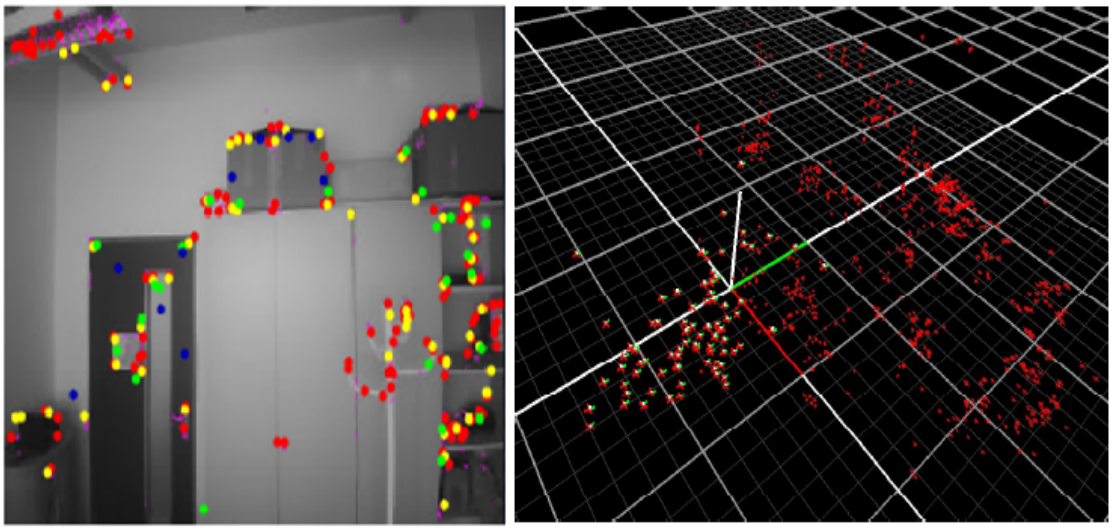
$$e(p, x_W, C) = \tilde{p} - proj(\mathrm{K}_{cam} \mathrm{proj}(\mathrm{E}_C \tilde{\mathrm{x}}_W) \in \mathbb{R}^2 \qquad [81]$$

Que se corresponde con la distancia en pixeles entre el punto en el que el marcador fue realmente observado y su proyección en la imagen. El error de reproyección total, para todas las observaciones p_{ij}, todos los marcadores $x_1,\ldots,x_n$ y todos los keyframes $K_1,\ldots,\mathrm{K}_m$ va a ser:

$$E_{rep}(x_1,\ldots,x_n,K_1,\ldots,K_m) = \sum_{\substack{j=1,\ldots,m \\ i\in\ell_j}} Obj\left(\frac{\left|\left|e_{ij}\right|\right|^2}{\sigma_{ij}^2}\right) \qquad [82]$$

Donde $e_{ij} = e(p_{ij}, \mathrm{x}_i, K_j)$, Obj es una función kernel robusta de $\mathbb{R} \to \mathbb{R}$, y ℓ_j es el conjunto de índices de todos los marcadores observados en el keyframe j. Esta función error total se minimiza por un proceso iterativo conocido como Levenberg-Marquardt y se conoce como bundle adjustement (*BA*).

Resulta sin embargo obvio que optimizar esta función en conjunto cada vez que se añade un keyframe o marcador no es factible computacionalmente. Es por ello que se emplea el bundle adjustement de forma local, es decir, se selecciona un pequeño subgrupo de keyframes y de marcadores, manteniendo todo lo demás fijado. De este modo, cada vez que se añade un keyframe, se optimizan solo los más recientes, cogiendo un grupo reducido de marcadores, ya que se asume que los keyframes y marcadores más antiguos ya son los suficientemente precisos.



(a) *(b)*

***Figura 53**. (a) Representación de los keypoints que usa el PTAM para el proceso de seguimiento. (b) Aspecto del mapa 3D que crea el PTAM. Los marcadores se corresponden con los puntos rojos. Las cruces rojas, verdes y blancas representan los keyframes donde se observaron dichos marcadores [21].*

Es importante mencionar que cada vez que un nuevo keyframe es identificado en el bucle de seguimiento, se añade al grupo de keyframes ya conocidos empleando la posición de la cámara (conocida) y las observaciones de los marcadores existentes. Para generar nuevos marcadores, se extraen nuevos puntos de los nuevos frames y mediante un mecanismo de búsqueda epipolar se trata de buscar coincidencias de alguno de los nuevos marcadores con alguno de los que había antes. Si se encuentra alguna coincidencia, entonces la nueva posición del marcador puede ser triangularizada y se añade al mapa. Posteriormente se lleva a cabo un Bundle Adjustement de forma local.

1.1.3 Seguimiento

Este bucle se ejecuta una vez para cada nuevo frame de video I y calcula la correspondiente posición de la cámara C en base a la posición de los marcadores conocidos $(x_1,...,x_n)$. Se requiere una suposición inicial sobre cuál es la posición de partida de la cámara C_0, que puede asimilarse por ejemplo a la posición de la cámara en el anterior frame.

En primer lugar, todos los marcadores potencialmente visibles son proyectados en la imagen en base a la posición inicial estimada de la cámara C_0, y empleando el proceso de tracking descrito en apartado en el que se explica la inicialización se establece la localización exacta de cada marcador en la imagen. Es decir, en este paso se establecen k correspondencias de puntos de 3D a 2D: $(x_1,...,x_n) \rightarrow (p_1,...,\mathrm{p}_n)$.

En base a estas correspondencias, la posición *C* de la cámara puede ser estimada. Este problema se conoce como *PnP* (perspective n-point problem). Existen diversos métodos de resolver este problema pero en el software que nos ocupa, debido a que en los sistemas con SLAM puede asumirse por lo general que hay una buena inicialización disponible y el movimiento de la cámara es pequeño entre dos frames, se ha escogido un método basado en minimizar el error de reproyección:

$$C^* = \arg\min_C \sum_{i=1}^{k} Obj\left(\frac{\left\|e(p_i, x_i, C\right\|^2}{\sigma_i^2}\right) \qquad [83]$$

Cuando se pierde el proceso de seguimiento debido a algún problema (pérdida de contacto visual, etc.), el método descrito anteriormente no puede ser aplicado debido a que no tenemos una estimación inicial (C_0) de la posición de la cámara. Se necesita llevar a cabo un proceso de recuperación del seguimiento de forma independiente. Para ello se emplean unos sistemas llamados descriptores de perspectiva invariante como el SIFT [34] y el SURF [35]. El PTAM lo que hace es tratar de hacer la recuperación tratando de ver con cuál de los keyframes anteriormente conseguidos encaja mejor el frame actual. En base a este encaje, se hace una suposición inicial de la posición de la cámara.

El bucle del tracking también decide si un determinado frame será añadido como nuevo keyframe en base a criterios heurísticos como si la calidad del tracking es buena (se ha encontrado un elevado número de marcadores), no se ha añadido ningún keyframe durante algún tiempo o que ningún keyframe fue tomado desde un punto cercano a la posición actual de la cámara.

1.1.4 Estimación de la escala del mapa

Uno de los principales defectos de los sistemas genéricos de PTAM es que la escala del mapa $\lambda \in \mathbb{R}$ no puede ser determinada sin usar información adicional procedente de los sensores o conocimiento sobre objetos presentes en la escena como marcadores artificiales. Sin embargo, para realizar una navegación autónoma, estimar este factor de escala es esencial.

Para realizar la estimación de la escala se propone lo siguiente: supongamos que tenemos una secuencia continua de estimaciones de la posición dadas por el PTAM que denotamos como $p_v(t)$ y una secuencia de medidas con ruido de los sensores de la posición absoluta $p_s(t)$, la velocidad $v_s(t)$ o la aceleración $a_s(t)$. El método desarrollado requiere que $v_s(t)$ pueda ser medida sin deriva. En el caso del AR Drone: las velocidades horizontales son calculadas a bordo por el propio dron, usando un altímetro por ultrasonidos y una serie algoritmos de flujo ópticos aplicados sobre las imágenes obtenidas por la cámara inferior del dron. Debido a las discontinuidades y deriva que aparecen al utilizar estos sensores, la trayectoria recorrida se divide en intervalos para poder trabajar en un supuesto donde no se tenga deriva. Se consideran intervalos de tiempo regulares, siendo la distancia recorrida por el dron entre dos instantes de tiempo determinados:

$$x_i = p_v(t_i) - p_v(t_{i-1}) \qquad [84]$$

$$y_i = \int_{t_{i-1}}^{t_i} v_s(t)\,dt \qquad [85]$$

Si se asume que tenemos ruido gaussiano e independiente en todas las medidas, el ruido en x_i e y_i es también independiente y gaussiano. Además, se asume que: la varianza del ruido de x_i es una constante σ_x^2 e independiente de λ. Esto es así porque el ruido en $p_v(t)$ depende de muchos factores, pero fundamentalmente crece linealmente con la profundidad de los puntos observados. Como la escala inicial puede ser escogida de forma que la profundidad media sea constante, se puede asumir por tanto que el ruido va a ser aproximadamente constante, considerando que la profundidad media no cambia de forma significativa con el tiempo. Por otro lado, se asume que la varianza del ruido en y_i es una constante σ_y^2 lo que se deriva de que el ruido en $v_s(t)$ tiene una varianza constante. Es importante destacar que σ_y^2 crece linealmente con el tamaño del intervalo.

Se formula pues el problema de averiguar la escala como un problema estadístico, donde tenemos dos grupos de medidas correspondientes con ruido y d-dimensionales: $X = \{x_1,...,x_n\}$ e $Y = \{y_1,...,y_n\}$ que representan la distancia verdadera recorrida. Las muestras se considera que tienen una distribución $x_i : N(\lambda\mu_i, \sigma_x^2)$ e $y_i : N(\mu_i, \sigma_x^2)$ donde $\mu_i \in \mathbb{R}^d$ representa la distancia verdadera recorrida (que desconocemos) y σ_x^2 e $\sigma_y^2 \in \mathbb{R}^+$ son las varianzas conocidas de los errores en la medida.

Para resolver el problema, se sigue el planteamiento de máxima verosimilitud, aplicando una función logarítmica de verosimilitud negativa:

$$L(\mu_1,...,\mu_n,\lambda) \propto \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\left\|x_i-\lambda\mu_i\right\|^2}{\sigma_x^2},\frac{\left\|y_i-\mu_i\right\|^2}{\sigma_y^2}\right) \quad [86]$$

El estimador de máxima verosimilitud para λ se calcula como:

$$\lambda^* = \arg\min_{\lambda}\min_{\mu_1,...,\mu_n} L(\mu_1,...,\mu_n,\lambda) \quad [87]$$

Si hacemos la derivada de $L(\mu_1,...,\mu_n,\lambda)$ con respecto a μ_i e igualamos a cero podemos calcular los valores óptimos para μ_i en función de λ:

$$\frac{\partial L(\mu_1,...,\mu_n,\lambda)}{\partial \mu_i} \propto \frac{\lambda^2\mu_i-\lambda x_i}{\sigma_x^2}+\frac{\mu_i-y_i}{\sigma_y^2}=0 \quad [88]$$

$$\mu_i = \frac{\lambda\sigma_y^2 x_i-\sigma_x^2 y_i}{\lambda^2\sigma_y^2+\sigma_x^2} \quad [89]$$

Se puede plantear de este modo lo siguiente:

$$\lambda^* = \arg\min_{\lambda}\hat{L}(\lambda) \quad [90]$$

Donde:

$$\hat{L}(\lambda) = \min_{\mu_1,...,\mu_n} \mathrm{L}(\mu_1,...,\mu_n,\lambda) \propto \frac{1}{2}\sum_{I=1}^{n}\frac{\left\|x_i-\lambda y_i\right\|^2}{\lambda^2\sigma_y^2+\sigma_x^2} \quad [91]$$

Si de nuevo repetimos el proceso de derivación para $\hat{L}(\lambda)$ e igualamos a cero, podemos obtener el valor óptimo para λ:

$$\hat{L}'(\lambda) \propto \frac{1}{2}\sum_{i=1}^{n}\frac{\left(\lambda y_i-x_i\right)^T\left(\sigma_x^2 y_i+\lambda\sigma_y^2 x_i\right)}{\left(\lambda^2\sigma_y^2+\sigma_x^2\right)^2}=0 \quad [92]$$

Si se tienen en cuenta una serie de consideraciones, tales como que $\lim_{\lambda\to\infty}\hat{L}(\lambda)=\lim_{\lambda\to-\infty}\hat{L}(\lambda)\in\mathbb{R}$, qué $\exists\lambda_0\in\mathbb{R}^+:\forall\lambda>\lambda_0:\hat{L}'(\lambda)>0\wedge\hat{L}'(-\lambda)>0$ y que $\hat{L}(\lambda)$ tiene como máximo dos extremos se puede demostrar que $\hat{L}(\lambda)$ tiene un único mínimo global en:

$$\lambda^* = \frac{s_{xx}-s_{yy}+\sqrt{\left(s_{xx}-s_{yy}\right)^2+4s_{xy}^2}}{2\sigma_x^{-1}\sigma_y s_{xy}} \quad [93]$$

Con: $s_{xx} = \sigma_y^2 \sum_{i=1}^{n} x_i^T x_i$, $s_{yy} = \sigma_x^2 \sum_{i=1}^{n} y_i^T y_i$ y $s_{xy} = \sigma_x \sigma_y \sum_{i=1}^{n} x_i^T y_i$.

Dicho mínimo se corresponde con el estimador de máxima verosimilitud para el factor de escala λ . Como se desprende de la observación de la fórmula, las varianzas de los errores de medida σ_x^2, σ_y^2 tienen una influencia que no puede ser despreciada en la estimación de la escala. Se demuestra que es imprescindible conocer los valores de σ_x^2, σ_y^2 o al menos una estimación de los mismos para obtener un valor para λ . En caso contrario, solo pueden conocerse los límites superior e inferior dentro de los que está comprendido λ .

2. Filtro de Kalman Extendido

En el ámbito de la robótica, uno de los principales objetivos es ser capaz de controlar un sistema dinámico, es decir, un sistema que cambia su estado a medida que pasa el tiempo. En el caso que nos ocupa, lo que se desea manejar es un cuadricóptero, y su estado va a venir descrito por su posición, su orientación y su velocidad. Para poder controlar de forma precisa un sistema de estas características, se emplean sensores para recoger información con la cual poder inferir el estado actual del sistema de la forma más precisa posible. Sin embargo, las medidas que los sensores recogen están sujetas a errores, que es lo que se conoce generalmente como ruido. Es por eso que se hace indispensable un filtrado de las medidas. Del mismo modo, lo más común es que el robot disponga de múltiples sensores, que pueden ser utilizados para adquirir información sobre una misma variable de estado, de forma que combinando las diferentes medidas se pueda obtener una única estimación de mayor precisión.

Todo lo comentado constituye la explicación de porqué resulta imprescindible la implementación de un sistema de filtrado y fusión de datos. Una de las opciones más ampliamente utilizadas es lo que se conoce como filtro de Kalman. Este sistema, en su forma más simple, se presenta como un filtro lineal, que funciona bajo la suposición de que las variables del problema tienen una distribución gaussiana multivariante y que las medidas están sujetas a ruido gaussiano e independiente, que nos permite obtener una estimación del estado actual del sistema (cuadricóptero) así como la incertidumbre (covarianza) de la estimación.

El filtro de Kalman lineal opera según un bucle continúo en el que se realizan dos tareas: predicción y actualización del estado. La fase de predicción usa la estimación del instante de tiempo anterior (k-1) para producir una estimación del estado en el instante de tiempo actual (k). Esta etapa se conoce también como estimación de estado *a priori,* ya que a pesar de ser una estimación del estado actual del sistema, no incluye la información aportada por la observación en el instante de tiempo actual. La fase de actualización por el contrario, se conoce como estimación *a posteriori*, ya que la

predicción *a priori* se combina con la información de la observación actual, refinándose el resultado de la estimación. Generalmente las dos fases se alternan, con la predicción avanzando el estado hasta que la siguiente observación programada se realice, y la actualización incorporando la observación y refinando la estimación de estado. Sin embargo, no es necesario que los pasos se produzcan sucesivamente, ya que por ejemplo, si por algún motivo no tenemos una observación disponible, la fase de actualización puede ser saltada y llevarse a cabo múltiples pasos de predicción. A continuación detallamos unas definiciones previas y la formulación matemática del filtro:

- n es la dimensión del vector de estado, m la dimensión del vector observación (que representa las medidas tomadas por los sensores) y d es la dimensión del vector de control.
- $x_k \in \mathbb{R}^n$ es el vector de estado verdadero en el instante de tiempo k. La estimación de este vector, incorporando las medidas hasta el instante de tiempo j (este inclusive) se denota como $\hat{x}_{k|j}$.
- $P_{k|j} \in \mathbb{R}^{n \times n}$ es la covarianza estimada de $\hat{x}_{k|j}$
- $B \in \mathbb{R}^{n \times d}$ es una matriz que representa el modelo del sistema de control, permitiendo representar el efecto que el vector de control $u_k \in \mathbb{R}^d$ en el estado interno.
- $F \in \mathbb{R}^{n \times n}$ es una matriz que representa el modelo de transición de estado, es decir, permite representar como cambia el estado del sistema de un tiempo k-1, a un tiempo k. Esta transición se considera que está sujeta a un ruido gaussiano de media cero $w_k : N(0,Q)$ donde Q es conocido. De este modo, resulta que: $x_k = F\, x_{k-1} + B\, u_k + w_k$.
- $H \in \mathbb{R}^{m \times n}$ es una matriz que representa el modelo de observación, permitiendo transformar el estado interno del sistema en la observación que le correspondería.
- $z_k \in \mathbb{R}^m$ es la observación en el tiempo k. Esta observación se considera que está sujeta a un ruido gaussiano de media cero $v_k : N(0,\mathrm{R})$ donde R es conocida. De este modo, resulta que: $z_k = H\, x_k + v_k$.

1. Predicción

Estimación *a priori* del estado: $\hat{x}_{k|k-1} = F\, \hat{x}_{k-1|k-1} + B\, u_k$ [94]

Estimación *a priori* de la covarianza: $P_{k|k-1} = F\, \mathrm{P}_{k-1|k-1}\, F^T + Q$ [95]

2. Actualización

Innovación o residuo con respecto a la medida: $y_k = z_k + H\, \hat{x}_{k|k-1}$ [96]

Innovación o residuo con respecto a la covarianza: $S_k = H\ P_{k|k-1}H^T + R$ [97]

Ganancia óptima Kalman: $K_k = P_{k|k-1}H^T\ S_k^{-1}$ [98]

Estimación *a posteriori* del estado: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$ [99]

Estimación *a posteriori* de la covarianza: $P_{k|k} = \left(I - K_k H\right)P_{k|k-1}$ [100]

Una vez presentado el filtro de Kalman lineal podemos explicar el filtro de Kalman extendido, que es el que realmente está implementado en el software TUM AR Drone, ya que constituye una variante del anterior pero con la característica fundamental de que es aplicable a sistemas no lineales, que es lo característico de los sistemas reales. La única diferencia con respecto al filtro lineal es que los modelos para la transición de estado y las observaciones ya no tienen por qué ser funciones lineales, sino que ahora más bien se plantean como dos funciones diferenciables $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$ y $h : \mathbb{R}^n \to \mathbb{R}^m$ tales que:

$$\hat{x}_k = f\left(x_{k-1}, u_k\right) + w_{k-1} \quad [101]$$

$$z_k = h\left(x_k\right) + v_{k-1} \quad [102]$$

La principal dificultad que se presenta en este caso es que cuando aplicamos una transformación no lineal a una variable aleatoria gaussiana, la variable aleatoria resultante ya no es gaussiana. Para poder seguir utilizando la estructura explicada para el filtro lineal, tenemos que aplicar un desarrollo de Taylor de primer orden a las funciones *f* y *h*, resultando:

$$F_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1}, u_k} \quad [103]$$

$$H_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k|k-1}} \quad [104]$$

Debemos tener en cuenta pues, que a diferencia de lo que ocurría con el filtro lineal, en este caso *H* y *F* van a ser diferentes para cada instante. El proceso de predicción y actualización es como sigue:

1. Predicción

Estimación *a priori* del estado: $\hat{x}_{k|k-1} = f\ \left(\hat{x}_{k-1|k-1}, u_k\right)$ [105]

Estimación *a priori* de la covarianza: $P_{k|k-1} = F_{k-1}\ \mathrm{P}_{k-1|k-1}\ F_{k-1}^T + Q$ [106]

2. Actualización

Innovación o residuo con respecto a la medida: $y_k = z_k + h\left(\hat{x}_{k|k-1}\right)$ [107]

Innovación con respecto a la covarianza: $S_k = H_k \; P_{k|k-1} H_k^T + R$ [108]

Ganancia óptima Kalman: $K_k = P_{k|k-1} H_k^T \; S_k^{-1}$ [109]

Estimación *a posteriori* del estado: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$ [110]

Estimación *a posteriori* de la covarianza: $P_{k|k} = \left(I - K_k H\right) P_{k|k-1}$ [111]

3. Controlador PID

La teoría del control automático trata sobre cómo resolver el problema de controlar el comportamiento de un sistema dinámico, es decir, de un sistema cuyo estado cambia con el tiempo. El objetivo a cumplir es calcular cuál tiene que ser la entrada del sistema $u(t)$ para que este alcance y mantenga el estado deseado. En otras palabras, el error $e(t)$ entre un determinado objetivo $w(t)$ y la salida del sistema y(t) debe ser minimizado a medida que pasa el tiempo.

El mecanismo de control utilizado para controlar el AR Drone es un controlador Proporcional Integral Derivativo (PID), que es un lazo de control ampliamente usado para control de sistemas industriales (ver figura 54). A continuación explicamos que es cada uno de los términos:

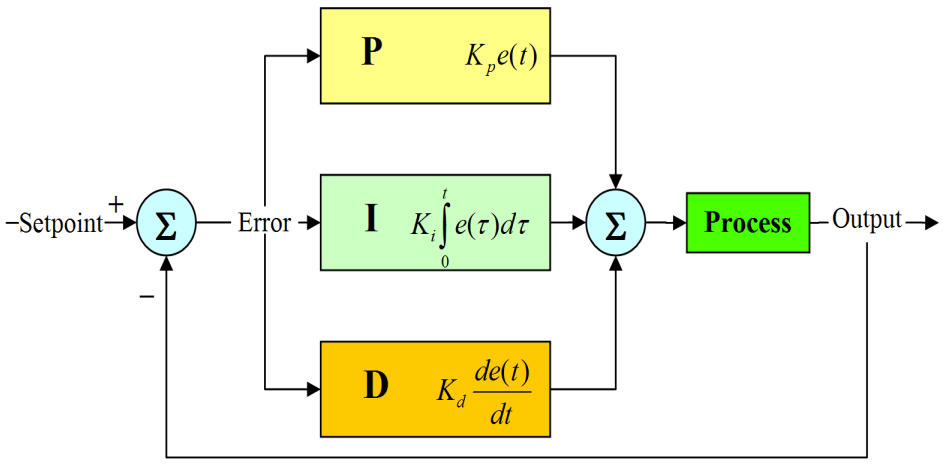


Figura 54. *Esquema general de un controlador PID [28].*

Parte proporcional

Esta parte depende del error actual $e(t)$ y siempre se requiere que aparezca, ya que es la parte responsable de que se reduzca el error. Cuanto mayor sea el error, más fuerte será

la señal de control aplicada. A la hora de controlar sistemas reales, la implementación de un controlador puramente proporcional no suele funcionar muy bien, debido a que provoca sobreexcitaciones y fuertes oscilaciones (ver figura 55.a).

Parte derivativa

Esta parte depende de la predicción del error futuro, en base a la derivada del error con respecto al tiempo $\dot{e}(t)$. La adición de este término tiene el efecto de amortiguar las oscilaciones: cuanto mayor sea el ritmo de cambio del error, más va a contribuir este término a reducir ese ritmo de cambio, reduciendo así sobreexcitaciones y oscilaciones (ver figura 55.b).

Parte integral

Esta parte depende del error pasado acumulado $\int_0^t e(\tau)\,\mathrm{d}\tau$.

Este término es el responsable de eliminar los errores de estado estacionario. En sistemas que requieren un control constante de la entrada que reciben para mantener su estado, un controlador puramente proporcional – derivativo (PD) hará que el sistema en lugar de permanecer en el estado deseado se quedará por encima o por debajo de ese valor debido al error acumulado. De todos modos, es importante tener en cuenta que el término integral debe tratarse con cuidado, puesto que puede hacer aumentar el tiempo de convergencia y causar fuertes oscilaciones (ver figura 55.c)

Si los términos integral y derivativo no pueden ser medidos directamente se aproximan por integración diferenciación numérica como:

$$\int_0^t e(\tau)d\tau \simeq \sum_{\tau=1}^{t} e(\tau) \qquad [112]$$

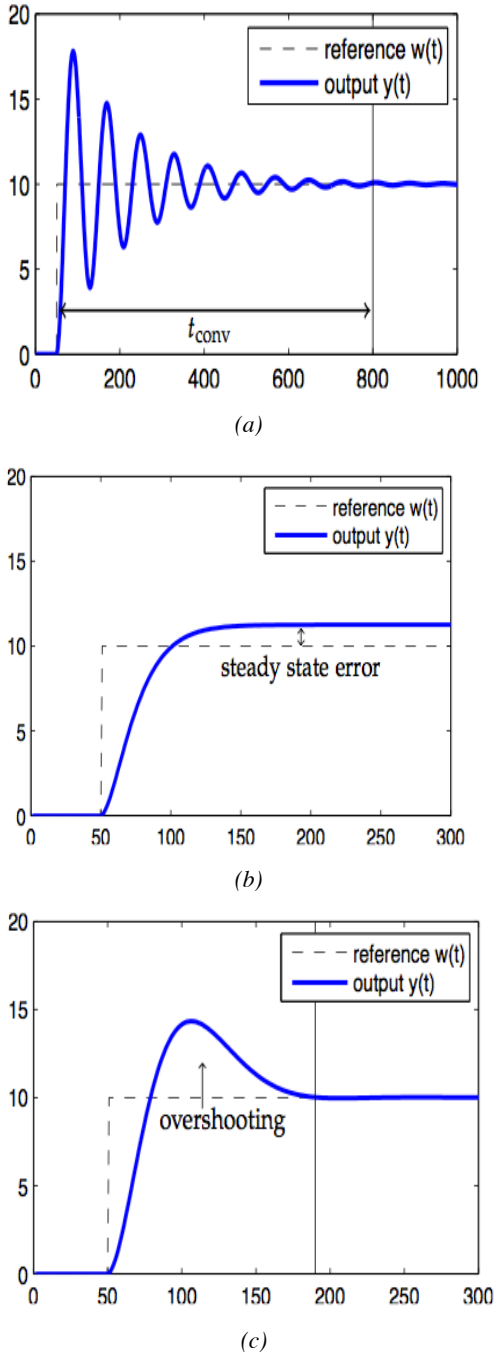
$$\dot{e}(t) \simeq \frac{1}{\delta_t}\left(e(t) - e\left(t-\delta_t\right)\right) \qquad [113]$$

La salida del sistema se calcula como:

$$u(t) = K_p\ e(\mathrm{t}) + K_i \int_0^t e(\tau)\ d\tau + K_d\ \dot{e}(t) \qquad [114]$$

Donde K_p, K_i, K_d son parámetros ajustables, que se suelen determinar experimentalmente por métodos de ensayo y error, aunque generalmente se siguen métodos heurísticos como el de Ziegler – Nichols. La calidad del sistema de control se determina por el tiempo de convergencia t_{conv}, es decir, el tiempo que se tarda en que $e(t)$ permanezca dentro de un pequeño intervalo determinado por el usuario alrededor de cero.

Este sistema implementado en el caso concreto del AR Drone envía comandos de control con una frecuencia de 100 Hz. En cada señal de control $\left(\bar{\Phi},\bar{\Theta},\bar{z},\bar{\Psi}\right)^T$ se define el roll, el pitch, la velocidad angular con respecto al eje z (velocidad de rotación del yaw) y la velocidad vertical (es decir en el eje z, que se da como fracción del valor máximo permitido $\bar{z}\in\left[-1,1\right]$).



(a)

(b)

(c)

Figura 55. *(a) Respuesta de un controlador de tipo P con respecto al tiempo. (b) Respuesta de un controlador de tipo PD con respecto al tiempo. (c) Respuesta de un controlador de tipo PID con respecto al tiempo [21].*

Al controlador se le proporciona como entrada la posición objetivo que deseamos alcanzar: $p = \left(\hat{x}, \hat{y}, \hat{z}, \hat{\Psi}\right)^T$ y el estado del cuadricóptero en el instante que le llegue la orden, es decir, el estado predicho por el filtro de Kalman en $t + \Delta t_{control}$ $T = (x, y, z, \Psi)$. Una vez se obtienen estos datos, se aplica control PID a los cuatro grados de libertad que se definen en el comando de control, rotándose el resultado en z para expresar el resultado en el sistema de coordenadas del cuadricóptero, dando cuenta de la posible orientación distinta de cero en dicho eje debido a algún giro del cuadricóptero. La matriz aplicada es la siguiente:

$$R(\Psi) = \begin{bmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{bmatrix} \qquad [115]$$

A continuación se detallan cuáles son las ecuaciones empleadas para determinar la salida del sistema $u(t)$ en cada uno de los distintos grados de libertad.

$$\begin{pmatrix} \bar{\Phi} \\ \bar{\Theta} \\ \bar{\dot{z}} \\ \bar{\dot{\Psi}} \end{pmatrix} = \begin{bmatrix} R(\Psi)\begin{bmatrix} 0.5\ (\hat{x} - x) + 0.32\ \dot{x} + 0 \\ 0.5\ (\hat{y} - y) + 0.32\ \dot{y} + 0 \end{bmatrix} \\ 0.6\left(\hat{z} - z\right) + 0.2\ \hat{\dot{z}} + 0.01\int\left(\hat{z} - z\right) \\ 0.02\left(\hat{\Psi} - \Psi\right) + 0 + 0 \end{bmatrix} \qquad [116]$$

Para $\hat{x}$ e $\hat{y}$ se emplea un controlador PD con unas ganancias de control óptimas determinadas experimentalmente de valor $K_p = 0.5,\ K_d = 0.32,\ K_i = 0$. Se ha determinado que el término integral no es necesario, ya que el error que se acumula es prácticamente nulo, y por tanto, no se generan problemas de desviación de la posición deseada en estas coordenadas en estado estacionario.

Para $\hat{z}$ se emplea un controlador PID con unas ganancias de control óptimas determinadas experimentalmente por los investigadores de la Universidad Técnica de Múnich, tomando un valor de $K_p = 0.6,\ K_d = 0.2, K_i = 0.01$. El término integral se resetea cuando la posición establecida en el target (p) en esta coordenada se alcanza, estando limitado a un valor máximo de 0.2.

Para Ψ se emplea simplemente un controlador proporcional, ya que experimentalmente se ha determinado que no se producen oscilaciones ni sobreexcitaciones, así como tampoco desviaciones en estado estacionario por acumulación de errores. Las ganancias de control óptimas fueron determinadas también experimentalmente por los investigadores, y toman un valor de $K_p = 0.02,\ K_d = 0, K_i = 0$.

Anexo III : April Tags Fiducial System.

El objetivo de este anexo es explicar de forma detallada el funcionamiento de las librerías del April Tags Fiducial System, que se compone de cuatro procesos básicos: la detección de los segmentos de línea, presentado en el apartado 1, la detección de quads, que se presenta en el apartado 2, homografía y estimación extrínseca en el apartado 3 y la decodificación de la información en el apartado 4. También se ha incluido un quinto apartado donde se profundiza más en como funciona el sistema de codificación.

1. Detección de los segmentos de línea

El funcionamiento de los April Tags empieza con la detección de las líneas de la imagen. Para ello se calcula el gradiente en dirección y magnitud de cada pixel para luego ir aglomerando estos en grupos de pixeles con un gradiente similar. Este proceso de agrupación se hace de forma similar al método gráfico de Felzenszwalb [22]: se crea un gráfico en el que cada nodo representa un pixel. Los bordes se añaden entre pixeles adyacentes con un peso igual a la diferencia de estos pixeles en la dirección del gradiente. Estos bordes son clasificados y procesados en función a su peso de forma creciente: para cada borde, se prueba si los componentes a los que pertenecen los pixeles deberían ser unidos.

Dado un componente *n*, se denota el rango de las direcciones del gradiente como $D(n)$ y el rango de magnitudes como $M(n)$. Puesto de otro modo, $D(n)$ y $M(n)$ son valores escalares que representan la diferencia entre el valor máximo y mínimo de la dirección y magnitud del gradiente. En el caso de $D(n)$, habría que tener cuidado a la hora de manejar valores de en torno a 2π porque podría llevar a engaños. Sin embargo, como la mayoría de los bordes útiles tienen un valor mucho menor que π, no se presentan problemas serios.

Dados dos componentes *n* y *m*, estos se unirán si se cumplen las siguientes dos condiciones:

$$D(n \cup m) \leq \min\left(D(n), D(m)\right) + K_D / \left|n \cup m\right| \qquad [117]$$

$$M(n \cup m) \leq \min\left(M(n), M(m)\right) + K_M / \left|n \cup m\right| \qquad [118]$$

La interpretación intuitiva de estas expresiones matemáticas es la siguiente: los valores pequeños de $D(\,)$ y $M(\,)$ indican componentes con poca intravariación entre componentes. Dos grupos de pixeles van a ser unidos si su unión es aproximadamente tan uniforme como los grupos por separado. Un incremento pequeño en la intravariación entre los componentes se permite a través de los parámetros K_D y K_M,

parámetros que se hacen cada vez más pequeños a medida que los componentes se hacen más grandes, como es lógico. Los autores del sistema han establecido unos valores de $K_D = 100$ y $K_M = 1200$.

Por razones de coste computacional, los pesos de borde se cuantifican y almacenan como números de punto fijo. Esto permite a los bordes ser clasificados usando un algoritmo de clasificación lineal en el tiempo. La operación de unión de grupos de pixeles es llevada a cabo de forma eficiente por el algoritmo de búsqueda – unión, con los límites de dirección y magnitud del gradiente almacenados en un sencillo array. Este método de unión de grupos de píxeles basado en el gradiente es sensible al ruido en la imagen. Incluso pequeñas cantidades de ruido provocan una variación local de la dirección del gradiente, inhibiendo el crecimiento de la variación de los componentes. Para eliminar este ruido el sistema se procesa la imagen a través de un filtro paso bajo. A diferencia de otros ámbitos donde el filtrado puede distorsionar información importante de la imagen, los bordes de la imagen son intrínsecamente características de gran escala, de forma que el filtrado no provoca pérdida de información.

Una vez se lleva a cabo la operación de agrupamiento de los píxeles, los segmentos de línea se encajan a cada uno de los componentes conectados, generalmente empleando un proceso de minimización por mínimos cuadrados, ponderando cada punto con la magnitud del gradiente. Se ajusta cada segmento de línea de tal forma que el lado oscuro de la línea quede a la izquierda y el lado más iluminado a la derecha. El algoritmo de segmentación es la fase más lenta en el esquema de detección. Opcionalmente se puede llevar a cabo la segmentación a la mitad de la resolución de imagen, resultando en un aumento de la velocidad en un factor de 4. La operación de subsampleado puede ser combinada de forma eficiente con el filtro paso bajo. La consecuencia de este proceso de optimización es que se reduce un poco el rango de detección.

2. Detección de los “quads”

Una vez alcanzado este punto se tiene un grupo de segmentos de línea que han sido obtenidos de la imagen y ahora la tarea es encontrar secuencias de los mismos que generen una figura de cuatro lados, es decir, lo que se denominará como “quad”. La dificultad reside ahora en lograr hacer esto de la forma más robusta posible, es decir, que el algoritmo sea capaz de funcionar sin verse muy afectado por posible ruido o a que quede tapado parcialmente algún segmento de línea.

El enfoque que se escoge se basa en un mecanismo de búsqueda recursiva de profundidad: cada nivel del árbol de búsqueda añade un borde al “quad”. En la profundidad uno, se considerando todos los segmentos de línea. En las profundidades dos y cuatro, se consideran todos los segmentos que empiecen lo suficientemente cerca

de donde acabó el anterior segmento de línea, realizándose esta búsqueda en sentido contrario a las agujas del reloj. Este algoritmo logra su robustez característica frente a errores de segmentación y a al tapado de segmentos de línea mediante el ajuste del umbral de proximidad entre las líneas: cuanto mayor sea este umbral, mayores saltos en torno a los bordes pueden ser manejados. El umbral que está implementado es dos veces la longitud de la línea más cinco píxeles adicionales. El escoger este umbral implica que va a ver una baja tasa de resultados negativos y una alta tasa de resultados válidos.

También se incorpora una tabla de búsqueda para acelerar la búsqueda sobre si un segmento de línea comienza cerca de un punto en el espacio. Con esta optimización, junto con las herramientas de rechazar aquellos segmentos que no cumplan el seguir el sentido antihorario o que estén repetidos, el algoritmo de detección de "quads" representa una pequeña fracción del total de requerimientos de cálculo.

Una vez las cuatro líneas hayan sido encontradas, se crea un candidato para "quad". Las esquinas de esta figura son las intersecciones de las líneas que lo forman. Debido a que estas líneas se ajustan empleando datos de muchos pixeles, estas estimaciones de las esquinas son altamente precisas, del orden de una pequeña fracción de pixel.

3. Homografía y estimación extrínseca

Se calcula la matriz de homografía 3x3 que proyecta los puntos 2D en coordenadas homogéneas del sistema de coordenadas del marcador (donde $[0\ 0\ 1]^T$ representa el centro del mismo, extendiéndose una unidad en las direcciones x e y) al sistema de coordenadas de la imagen. La homografía se calcula usando un algoritmo conocido como Transformación Linear Directa (DLT del inglés Direct Linear Transform). Es importante decir que este método hace la transformación de forma dependiente de la escala.

El cálculo de la posición y orientación del marcador requiere información adicional: la longitud focal de la cámara y el tamaño físico del marcador, variables que deben ser declaradas en el programa en cuestión. Además, las matriz de homografía puede ser escrita como el producto de una matriz *P* de proyección de la cámara de dimensiones 3x4, que se supone conocida y lo que se denomina la matriz extrínseca truncada *E*, de dimensiones 4x3. La matriz *P* representa como hace la cámara la proyección de los puntos 3D del sistema mundo al plano 2D de la imagen. Según el modelo de cámara pinhole (estenopeica), esta transformación de la que hablamos se haría del siguiente modo:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \qquad [119]$$

Donde (x_1, x_2, x_3) son las coordenadas 3D de un punto arbitrario *p* relativas a un sistema de coordenadas centrado en la cámara; (y_1, y_2) las coordenadas de dicho punto en la imagen resultante; y f las distancia focal de la cámara. Se considera que tanto f como x_3 son siempre positivos. Para obtener la matriz de la cámara, esta expresión se reescribe en términos de coordenadas homogéneas: en lugar del vector (y_1, y_2) se considera el elemento proyectivo $y = (y_1, y_2, 1)$, y en lugar de una igualdad consideramos una igualdad dependiente de una escala, lo que se denota por ~ :

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \\ \frac{x_3}{f} \end{pmatrix} \sim \begin{pmatrix} x_1 \\ x_2 \\ \frac{x_3}{f} \end{pmatrix} \qquad [120]$$

Si se hace una representación de las coordenadas tridimensionales en coordenadas homogéneas x la matriz de proyección tendrá el siguiente aspecto:

$$\begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \rightarrow y \sim Px \qquad [121]$$

La matriz *P* de proyección de la cámara puede definirse por tanto como:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} \sim \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad [122]$$

Esta matriz describe la posición de la cámara en el sistema mundo, así como la dirección a la que esta apunta. Viene a ser similar a una matriz de transformación ya que está compuesta de una matriz de rotación *R* y un vector de translación *t*, sin embargo, no se corresponde exactamente con la rotación y translación de la cámara. La matriz tiene la siguiente forma:

$$E = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}$$

$$E = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad [123]$$

Donde *t* puede interpretarse como la posición del origen del sistema mundo en las coordenadas de la cámara, y las columnas de *R* representan las direcciones de los ejes del sistema mundo en el sistema de coordenadas de la cámara. Resumiendo, esta matriz extrínseca lo que hace es describir como el sistema mundo se transforma a las coordenadas de la cámara.

Como se puede observar en las fórmulas, esta matriz es típicamente de 4x4, sin embargo, cualquier posición en el marcador va a estar en $z = 0$ con respecto al sistema de coordenadas de este, por lo que podemos eliminar la tercera columna de la matriz extrínseca, dando lugar a lo que se conoce como matriz extrínseca truncada. Consecuentemente, con este planteamiento podemos reescribir cualquier coordenada del marcador como un punto homogéneo en 2D (con la consideración implícita de que *z* vale cero). La matriz de homografía puede plantearse pues como sigue:

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} = s\ P\ E$$

$$s\ P\ E = s \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & T_x \\ R_{10} & R_{11} & T_y \\ R_{20} & R_{21} & T_z \\ 0 & 0 & 1 \end{bmatrix} \quad [124]$$

Hay que destacar que no se puede resolver directamente para *E* debido a que *P* es deficiente en rango. Las ecuaciones desarrolladas van a quedar de la forma siguiente:

$$\begin{aligned} h_{00} &= s\, R_{00}\, f_x & h_{10} &= s\, R_{10}\, f_y & h_{20} &= s\, R_{20} \\ h_{01} &= s\, R_{01}\, f_x & h_{11} &= s\, R_{11}\, f_y & h_{21} &= s\, R_{21} \\ h_{02} &= s\, \mathrm{T}_x\, f_x & h_{12} &= s\, \mathrm{T}_y\, f_y & h_{22} &= s\, \mathrm{T}_z \end{aligned} \qquad [125]$$

Todas estas ecuaciones son fácilmente resolubles para los elementos R_{ij} y T_k excepto para el factor desconocido de escala *s*. Sin embargo, ya que las columnas de una matriz de rotación tienen por definición magnitud unidad, se pueden establecer unas restricciones iniciales para s. Además, como tenemos dos columnas de la matriz de rotación, podemos calcular el factor de escala como la media geométrica de sus magnitudes. El signo de este factor puede ser hallado si establecemos la condición de que el marcador debe aparecer en frente de la cámara, es decir, que $T_z < 0$. Por otro lado, la tercera columna de la matriz de rotación puede ser obtenida de forma sencilla calculando el producto vectorial de las dos columnas conocidas, ya que sabemos que las columnas de una matriz de rotación deben ser ortonormales. Hay que tener en cuenta no obstante, que el procedimiento DLT y la normalización que se efectúan no garantizan que la matriz de rotación sea estrictamente ortonormal. Para corregir esto, se calcula la descomposición polar de *R* minimizando la norma de la matriz de Frobenius del error.

4. Decodificación de la información

La última tarea que se ha de llevar a cabo es la lectura de los bits de información que constituyen el marcador. Esto se lleva a cabo calculando las coordenadas relativas al marcador de cada una de estos bits, transformándolos en coordenadas de imagen usando la homografía, para posterior segmentación de los píxeles resultantes por valor umbral. Buscando que le método sea robusto a cambios de iluminación (la cual puede variar no solo de un marcador a otro, sino que también par uno de un mismo tipo debido a las condiciones de luz exterior, pudiendo estar por ejemplo un lado del marcador más iluminado que otro) se usa un umbral dependiente de la situación espacial.

Concretamente, el sistema se ha implementado un modelo de variación espacial de la intensidad de los pixeles “negros” y un segundo modelo para los pixeles “blancos”. El modelo tiene la forma siguiente:

$$I(x,y) = Ax + Bxy + Cy + D \qquad [126]$$

Este modelo tiene cuatro parámetros que son fácilmente calculables usando una regresión por mínimos cuadrados. Se han construido como se ha comentado dos modelos: una para los pixeles negros, otro para los blancos. El umbral usado a la hora de decodificar los bits de datos es la media de los valores de intensidad predichos para los colores negro y blanco.

5. Comentarios sobre el sistema de codificación

Una vez se ha descodificado la información contenida en los quads, hay que aplicar un sistema de codificación que determine si dicha información es válida o no. Un buen sistema de este tipo debe cumplir una serie de requisitos: que maximice el número de códigos que es capaz de distinguir y el número de bits erróneos que es capaz de detectar y corregir, y que minimice el número de posibles confusiones entre un tag y otro o con otro elemento que no es un tag (falsos positivos) y el número total de bits que se requiere para identificar un tag. Cumplir todos estos requisitos es difícil, ya que normalmente la mejora de uno entre en conflicto con otros por lo que hay que buscar una solución de compromiso.

El sistema de codificación utilizado es lo que se conoce como lexicódigo, que está parametrizado por dos elementos: el número de bits n en cada palabra de código y la distancia mínima de Hamming d entre dos palabras cualquiera de código. Este último concepto se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra válida distinta perteneciente al mismo código. Cuanto mayor sea esta diferencia, menor es la posibilidad de que un código válido se transforme en otro código válido por acumulación de una serie de errores. Los lexicódigos son capaces de detectar $d/2$ y de corregir $(d-1)/2$ bits erróneos. Los April Tag se denotan por el número de bits codificados y por la distancia de Hamming mínima. Por ejemplo, el marcador 36h10 tiene una codificación de 36 bits y una distancia de Hamming mínima de 10.

Los lexicódigos funcionan según un criterio heurístico bastante sencillo: las palabras de código potencialmente válidas se consideran en orden lexicográfico (es decir de la más pequeña a la más grande), añadiéndose nuevas palabras al libro de código cuando están al menos a una distancia d de cualquier otra palabra previamente añadida. En el caso de sistemas visuales como los April Tags, el sistema de codificación debe ser robusto a la rotación, lo que implica que si un tag es rotado, 45, 90 o 270 grados tiene que seguir manteniendo una distancia de Hamming de d. Los algoritmos tradicionales de generación de códigos lexicográficos no tienen está propiedad, sin embargo, los creadores de los April Tags han añadido a dichos algoritmos nuevas restricciones que garanticen que esto se cumpla.

Hay que tener cuidado de todos modos, ya que existen algunas palabras de código que a pesar de satisfacer la distancia mínima de Hamming no son buenas elecciones. Por ejemplo, una palabra que consiste en todo ceros (lo que equivaldría a un tag en forma de un único cuadrado negro) sería un patrón geométrico que aparecen de forma espontánea en nuestro entorno, por lo que sería muy común que se diesen un gran número de falsos positivos (es decir, confusiones). Es por ello que el lexicódigo desarrollado rechaza aquella palabras que puedan vincularse a patrones geométricos sencillos. Se establece

una métrica basada en el número de rectángulos necesarios para generar un modelo 2D del tag: por ejemplo, un rectángulo sólido de color homogéneo como el mencionado necesitaría simplemente un rectángulo para ser modelizado, mientras que una tira de color negro, blanco y negro requeriría dos triángulos: un gran rectángulo negro con un pequeño rectángulo blanco intercalado. El sistema empleado con los April Tags está basado en tags de alta complejidad, que requieren de muchos rectángulos para ser reconstruidos, ya que es muy poco probable que aparezcan modelos semejantes de forma espontánea en el entorno, y por tanto resulta en tasas de falsos positivos mucho menores. Se establece que para que una detección pueda ser considerada un tag hacen falta como mínimo 10 rectángulos. Los tags con una complejidad menor son rechazados. Los resultados experimentales llevados a cabo en la investigación corroboran este planteamiento.

Índice de figuras

Índice de tablas

Bibliografía

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors", *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[2] Q. Lindsey, D. Mellinger and V. Kumar, "Construction of cubic structures with quadrotor teams", *Proc. of Robotics: Science and Systems (RSS)*, 2011.

[3] R. Ritz, M. Mueller and R. D'Andrea, "Cooperative quadrocopter ball throwing and catching", *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[4] Kushleyev, D. Mellinger and V. Kumar, "Towards a swarm of agile micro quadrotors", *Proc. of Robotics: Science and Systems (RSS)*, 2012.

[5] S. Grzonka, G. Grisetti and W. Burgard, "Towards a navigation system for autonomous indoor flying", *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2009.

[6] S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera", *Proc. IEEE Intl. Symposium of Robotics Research (ISRR)*, 2011.

[7] M. Achtelik, A. Bachrach, R. He, S. Prentice and N. Roy, "Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments", *Proc. SPIE Unmanned Systems Technology XI*, 2009.

[8] K. Schmid, F. Ruess, M. Suppa and D. Burschka, "State estimation for highly dynamic flying systems using key frame odometry with varying time delays", *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[9] V. Grabe, H. Bulthoff and P. Giordano, "Robust optical- flow based self-motion estimation for a quadrotor UAV", *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[10] T. Krajník, V. Vonásek, D. Fiser and J. Faigl, "AR-drone as a platform for robotic research and education", *Proc. Communications in Computer and Information Science (CCIS)*, 2011.

[11] D. Eberli, D. Scaramuzza, S. Weiss and R. Siegwart, "Vision based position control for MAVs using one single circular land- mark", *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 495 – 512, 2011.

[12] M. Achtelik, M. Achtelik, S. Weiss and R. Siegwart, " onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments", *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[13] S. Yang, S. A. Scherer and A. Zell, "An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle", *Journal of*

Intelligent and Robotic Systems, vol. 69, pp. 499 – 515, 2013.

[14] C. Bills, J. Chen and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues", *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[15] W. S. Ng and E. Sharlin, "Collocated interaction with flying robots", *Proc. IEEE Intl. Symposium on Robot and Human Interactive Communication*, 2011.

[16] J. Engel, J. Sturm and D. Cremers, "Camera-based navigation of a low-cost quadrocopter", *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[17] E. Olson, "AprilTag: A robust and flexible visual fiducial system", *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[18] A. Richardson, J. Strom, E. Olson, "AprilCal: Assisted and repeatable camera calibration", *Proc. IEEE Intl. Conf. on Intelligent Robot Systems (IROS),* 2013.

[19] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces", *Proc. IEEE Intl. Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection", *Proc. of the European Conference on Computer Vision (ECCV)*, 2006.

[21] J. Engel, "Autonomous Camera-Based Navigation of a Quadrocopter," Master's thesis, Technical University Munich, 2011.

[22] Pedro F. Felzenszwalb, "Efficient Graph-Based Image Segmentation", *International Journal of Computer Vision,* vol. 59, pp. 167 – 181, 2004.

[23] Maja J. Mataric, "The Robotics Primer", *MIT Press,* 2007.

[24] Jason M. O'Kane, "A Gentle Introduction to ROS", *CreateSpace Independent Publishing Platform, 2013*

[25] Aaron Martinez, Enrique Fernández, "Learning ROS for Robotics Programming", *Packt Publishing*, 2013.

[26] Robin R. Murphy, "An Introduction to AI Robotics", *A Bradford Book*, 2000

[27] *http://www.amazon.com.*

[28] *http://www.wikipedia.org*

[29] *https://www.edx.org/course/tumx.*

[30] *http://www.edwardrosten.com.*

[31] *http://ardrone2.parrot.com/*

[32] *http://www.solver.com/smooth-nonlinear-technology*

[33] *http://opencv.org/*

[34] *http://en.wikipedia.org/wiki/Scale-invariant_feature_transform*

[35] *http://en.wikipedia.org/wiki/SURF*